

? logon

\*\*\* It is now 2011/11/20 16:36:51 \*\*\*  
(Dialog time 2011/11/20 16:36:51)

705TEXT1 is set ON as an alias for 15, 16, 160, 148, 621, 275, 634, 47  
705TEXT2 is set ON as an alias for 9, 623, 810, 624, 813, 20, 636  
705BIBLIT is set ON as an alias for 77, 35, 583, 2, 65, 233, 99  
705NEWSBIB is set ON as an alias for 473, 474, 475  
SOFTLIT is set ON as an alias for 256, 278  
705ADLIT is set ON as an alias for 635, 570, PAPERSMJ, PAPERSEU  
HIGHLIGHT set on as '' ''  
DETAIL set off  
KNIC is set to 50.  
FTEXT1 is set ON as an alias for 15,9,610,810,275,634,471  
FTEXT2 is set ON as an alias for 613,813,636,16,160,621,148,20,624  
FTPAT is set ON as an alias for 348,65  
NFTEXT is set ON as an alias for 35,65,99,2,583,474,475,347,256

? b

**15,9,610,810,275,624,621,636,613,813,16,160,634,148,20,35,583,65,2,474,475,99,256,635,570,  
47,608,PAPERSMJ,PAPERSEU**

20nov11 16:37:39 User264751 Session D859.1  
\$0.00 0.249 DialUnits File415  
\$0.00 Estimated cost File415  
\$0.22 INTERNET  
\$0.22 Estimated cost this search  
\$0.22 Estimated total session cost 0.249 DialUnits

SYSTEM:OS - DIALOG OneSearch  
File 15:ABI/Inform(R) 1971-2011/Nov 19  
(c) 2011 ProQuest Info&Learning  
File 9:Business & Industry(R) Jul/1994-2011/Nov 18  
(c) 2011 Gale/Cengage  
File 610:Business Wire 1999-2011/Nov 20  
(c) 2011 Business Wire.

\*File 610: contains data from 3/99 forward.  
For archive data (1986-2/99) see File 810.  
File 810:Business Wire 1986-1999/Feb 28  
(c) 1999 Business Wire

\*File 810: contains data from 1986-1999.  
See File 610 for current data.  
File 275:Gale Group Computer DB(TM) 1983-2011/Sep 27  
(c) 2011 Gale/Cengage  
File 624:McGraw-Hill Publications 1985-2011/Nov 18  
(c) 2011 McGraw-Hill Co. Inc  
File 621:Gale Group New Prod.Annou.(R) 1985-2011/Sep 16  
(c) 2011 Gale/Cengage  
File 636:Gale Group Newsletter DB(TM) 1987-2011/Nov 15  
(c) 2011 Gale/Cengage  
File 613:PR Newswire 1999-2011/Nov 20  
(c) 2011 PR Newswire Association Inc

\*File 613: File 613 now contains data from 5/99 forward.  
Archive data (1987-4/99) is available in File 813.  
File 813:PR Newswire 1987-1999/Apr 30

(c) 1999 PR Newswire Association Inc  
 \*File 813: contains data from 1987-1999.  
 For current data see File 613.

File 16:Gale Group PROMT(R) 1990-2011/Nov 15  
 (c) 2011 Gale/Cengage

File 160:Gale Group PROMT(R) 1972-1989  
 (c) 1999 The Gale Group

File 634:San Jose Mercury Jun 1985-2011/Nov 17  
 (c) 2011 San Jose Mercury News

File 148:Gale Group Trade & Industry DB 1976-2011/Nov 16  
 (c) 2011 Gale/Cengage

\*File 148: CURRENT feature not working. See HELP NEWS148.  
 File 20:Dialog Global Reporter 1997-2011/Nov 19  
 (c) 2011 Dialog

File 35:Dissertation Abs Online 1861-2011/Oct  
 (c) 2011 ProQuest Info&Learning

File 583:Gale Group Globalbase(TM) 1986-2002/Dec 13  
 (c) 2002 Gale/Cengage

\*File 583: This file is no longer updating as of 12-13-2002.  
 File 65:Inside Conferences 1993-2011/Nov 18  
 (c) 2011 BLDSC all rts. reserv.

File 2:INSPEC 1898-2011/Nov W2  
 (c) 2011 The IET

File 474:New York Times Abs 1969-2011/Nov 20  
 (c) 2011 The New York Times

File 475:Wall Street Journal Abs 1973-2011/Feb 14  
 (c) 2011 The New York Times

\*File 475: This file no longer updates. Please see  
 NewsRoom for most recent records.

File 99:Wilson Appl. Sci & Tech Abs 1983-2011/Oct  
 (c) 2011 The HW Wilson Co.

File 256:TecTrends 1982-2011/Apr W1  
 (c) 2011 Info.Sources Inc. All rights res.

\*File 256: TecTrends has suspended updates in April 2011.  
 File 635:Business Dateline(R) 1985-2011/Nov 19  
 (c) 2011 ProQuest Info&Learning

File 570:Gale Group MARS(R) 1984-2011/Nov 17  
 (c) 2011 Gale/Cengage

File 47:Gale Group Magazine DB(TM) 1959-2011/Oct 13  
 (c) 2011 Gale/Cengage

File 608:MCT Information Svc. 1992-2011/Oct 12  
 (c) 2011 MCT Information Svc.

File 387:The Denver Post 1994-2011/Nov 18  
 (c) 2011 Denver Post

File 471:New York Times Fulltext 1980-2011/Nov 20  
 (c) 2011 The New York Times

File 492:Arizona Repub/Phoenix Gaz 19862002/Jan 06  
 (c) 2002 Phoenix Newspapers

\*File 492: This file no longer updates. Last PD Jan. 2002.  
 NewsRoom contains newer records for some titles.

File 494:St LouisPost-Dispatch 1988-2011/Nov 18  
 (c) 2011 St Louis Post-Dispatch

File 631:Boston Globe 1980-2009/Dec 30  
 (c) 2010 Boston Globe

\*File 631: No longer updates, last PD=20091231. Please  
 see NewsRoom for current Boston Globe records

File 633:Phil.Inquirer 1983-2011/Mar 31



```
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processed 10 of 49 files ...
Processing
Processing
Processing
Processing
Processing
Processing
Processing
>>>One or more prefixes are unsupported
>>> or undefined in one or more files.
Processing
Processing
Processed 20 of 49 files ...
Processing
Processing
Processing
Processed 30 of 49 files ...
Processing
Processed 40 of 49 files ...
Processing
Completed processing all files
$1105294171 PD<20031017
```



**? s AU=(cirulli, s or cirulli s? or (susan(2n)cirulli) or BY=(susan(2n)cirulli)**

```
>>>One or more prefixes are unsupported
>>> or undefined in one or more files.
>>>Input error: Numeric characters expected
>>>Input error: Numeric characters expected
      0 AU=CIRULLI, S
      1 AU=CIRULLI S?
228357 AU=SUSAN
      9 AU=CIRULLI
      0 AU=SUSAN(2N)AU=CIRULLI
214 BY=SUSAN
      0 BY=CIRULLI
      0 BY=SUSAN(2N)BY=CIRULLI
S2      1 AU=(CIRULLI, S OR CIRULLI S? OR (SUSAN(2N)CIRULLI)) OR
      BY=(SUSAN(2N)CIRULLI)
```

**? s AU=(hager, d or hager d? or ((dan or danny)(2n)hager)) or BY=((dan or danny)(2n)hager)**

```
>>>One or more prefixes are unsupported
>>> or undefined in one or more files.
>>>Input error: Numeric characters expected
>>>Input error: Numeric characters expected
>>>Input error: Numeric characters expected
      0 AU=HAGER, D
      0 AU=HAGER D?
299586 AU=DAN
29504 AU=DANNY
1404 AU=HAGER
      4 (AU=DAN OR AU=DANNY) (2N)AU=HAGER
219 BY=DAN
      1 BY=DANNY
      0 BY=HAGER
      0 (BY=DAN OR BY=DANNY) (2N)BY=HAGER
S3      4 AU=(HAGER, D OR HAGER D? OR ((DAN OR DANNY) (2N)HAGER)) OR
      BY=((DAN OR DANNY) (2N)HAGER)
```

**? s s2 or s3**

```
      1 S2
      4 S3
S4      5 S2 OR S3
```

**? t s4/3/all**

4/3/1 (Item 1 from file: 20)  
DIALOG(R)File 20: Dialog Global Reporter  
(c) 2011 Dialog. All rights reserved.

46400380 (USE FORMAT 7 OR 9 FOR FULLTEXT)  
**THE ABRAMS REPORT For January 3, 2006, MSNBC - Part 2**

**Section Title:** News; Domestic  
**Dan Abrams, Bob Hager, Norah O'Donnell**  
MSNBC NEWS  
January 03, 2006  
**Journal Code:** WMSN **Language:** English **Record Type:** FULLTEXT  
**Word Count:** 3317

4/3/2 (Item 2 from file: 20)  
DIALOG(R)File 20: Dialog Global Reporter  
(c) 2011 Dialog. All rights reserved.

46400379 (USE FORMAT 7 OR 9 FOR FULLTEXT)  
**THE ABRAMS REPORT For January 3, 2006, MSNBC - Part 1**

**Section Title:** News; Domestic  
**Dan Abrams, Bob Hager, Norah O'Donnell**  
MSNBC NEWS  
January 03, 2006  
**Journal Code:** WMSN **Language:** English **Record Type:** FULLTEXT  
**Word Count:** 4689

4/3/3 (Item 3 from file: 20)  
DIALOG(R)File 20: Dialog Global Reporter  
(c) 2011 Dialog. All rights reserved.

33719818 (USE FORMAT 7 OR 9 FOR FULLTEXT)  
**THE NEWS ON CNBC for February 6, 2004, CNBC**

**Section Title:** News; Domestic  
**Ashleigh Banfield, Kerry Sanders, Dan Abrams, Robert Hager**  
CNBC NEWS  
February 07, 2004  
**Journal Code:** WCNC **Language:** English **Record Type:** FULLTEXT  
**Word Count:** 4330

4/3/4 (Item 4 from file: 20)  
DIALOG(R)File 20: Dialog Global Reporter  
(c) 2011 Dialog. All rights reserved.

30085042 (USE FORMAT 7 OR 9 FOR FULLTEXT)  
**THE NEWS WITH BRIAN WILLIAMS for June 26, 2003, CNBC - Part 1**

**Section Title:** News; International, News; Domestic  
**John Seigenthaler, Ron Allen, Jim Miklaszewski, Diana Olick, Pete Williams, Jim Cummins,**

Dan Abrams, Jim Cummins, **Dan Abrams**, Robert **Hager**, Mike Hegades  
 CNBC NEWS  
 June 06, 2003  
**Journal Code:** WCNC **Language:** English **Record Type:** FULLTEXT  
**Word Count:** 4427

4/3/5 (Item 1 from file: 35)  
 DIALOG(R)File 35: Dissertation Abs Online  
 (c) 2011 ProQuest Info&Learning. All rights reserved.

01453493 ORDER NO: AADAA-I9600076  
**A COMPARATIVE STUDY OF ACHIEVEMENT GAINS: LITERATURE-BASED VS  
 BASAL APPROACH TO JUNIOR HIGH READING**

**Author:** CIRULLI SCOBORIA, ROSEMARIE  
**Degree:** ED.D.  
**Year:** 1995  
**Corporate Source/Institution:** TEMPLE UNIVERSITY ( 0225 )  
**Source:** Volume 5609A of Dissertations Abstracts International.  
 PAGE 3527 . 82 PAGES

? ds

Set	Items	Description
S1	105294171	PD<20031017
S2	1	AU=(CIRULLI, S OR CIRULLI S? OR (SUSAN(2N)CIRULLI)) OR BY=-(SUSAN(2N)CIRULLI)
S3	4	AU=(HAGER, D OR HAGER D? OR ((DAN OR DANNY) (2N)HAGER)) OR -BY=((DAN OR DANNY) (2N)HAGER)
S4	5	S2 OR S3

? s s1 and ((DAN OR DANNY)(2N)HAGER) or (SUSAN(2N)CIRULLI))

>>>Unmatched parentheses

? s s1 and (((DAN OR DANNY)(2N)HAGER) or (SUSAN(2N)CIRULLI))

Processing

Processing	Items	Description
	105294171	S1
	2945005	DAN
	1122788	DANNY
	45540	HAGER
	159	((DAN OR DANNY) (2N)HAGER
	1897102	SUSAN
	305	CIRULLI
	0	SUSAN (2N) CIRULLI
S5	90	S1 AND (((DAN OR DANNY) (2N)HAGER) OR (SUSAN (2N) CIRULLI))

? s5 and ((approve or approver or approvers or approval or approvals or approving or signoff) and (purchase(w)(form or forms or order or orders or requestion or requisitions))

>>>Unmatched parentheses

? s5 and (approve or approver or approved or approvers or approval or approvals or approving) and (purchase(w)(form or forms or order or orders or requestion or requisitions))

```
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processed 10 of 49 files ...
Processing
Processing
Processed 20 of 49 files ...
Processing
Completed processing all files
          90 S5
        1730870 APPROVE
          3162 APPROVER
        7731809 APPROVED
          2840 APPROVERS
        5963781 APPROVAL
        1162822 APPROVALS
          411537 APPROVING
        9108164 PURCHASE
        15382052 FORM
        3680255 FORMS
        24126569 ORDER
        7756496 ORDERS
          34 REQUESTION
          9 REQUISITIONS
        199191 PURCHASE(W) (((((FORM OR FORMS) OR ORDER) OR ORDERS) OR
S6          0 S5 AND (APPROVE OR APPROVER OR APPROVED OR APPROVERS OR
              APPROVAL OR APPROVALS OR APPROVING) AND (PURCHASE(W) (FORM
              OR FORMS OR ORDER OR ORDERS OR REQUESTION OR
              REQUISITIONS))
```

? s1 and ((approve or approver or approved or approvers or approval or approvals or approving)(2n)(purchase(w)(order or orders or requestion or requisitions)))

```
Processing
Processing
Processing
```

```

Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processed 10 of 49 files ...
Processing
Processing
Processing
Processed 20 of 49 files ...
Processing
Processed 40 of 49 files ...
Completed processing all files
      105294171 S1
      1730870 APPROVE
      3162 APPROVER
      7731809 APPROVED
      2840 APPROVERS
      5963781 APPROVAL
      1162822 APPROVALS
      411537 APPROVING
      9108164 PURCHASE
      24126569 ORDER
      7756496 ORDERS
      34 REQUISTION
      9 REQUISITIONS
      2989 ((((((APPROVE OR APPROVER) OR APPROVED) OR APPROVERS) OR
      APPROVAL) OR APPROVALS) OR
      APPROVING) (2N)PURCHASE(W) (((ORDER OR ORDERS) OR
      REQUISTION) OR REQUISITIONS)
S7 1470 S1 AND ((APPROVE OR APPROVER OR APPROVED OR APPROVERS OR
      APPROVAL OR APPROVALS OR APPROVING) (2N) (PURCHASE(W) (ORDER
      OR ORDERS OR REQUISTION OR REQUISITIONS)))

```

**? s7 and (((dynamic or dynamically or recalculate or recalculates or recalculated or  
 recalculating or recalculation or recalculations)(4n)(approve or approves or approved or  
 approval or approvers or approval or approvals or approving))**

```

Processing
Processing
Processing
Processing
Processed 20 of 49 files ...
Completed processing all files
      1470 S7
      3141505 DYNAMIC
      316923 DYNAMICALLY
      18078 RECALCULATE
      3759 RECALCULATES
      22789 RECALCULATED

```

6884 RECALCULATING  
 13177 RECALCULATION  
 2908 RECALCULATIONS  
 1730870 APPROVE  
 629742 APPROVES  
 7731809 APPROVED  
 3162 APPROVER  
 2840 APPROVERS  
 5963781 APPROVAL  
 1162822 APPROVALS  
 411537 APPROVING  
 1553 (((((((DYNAMIC OR DYNAMICALLY) OR RECALCULATE) OR  
 RECALCULATES) OR RECALCULATED) OR RECALCULATING) OR  
 RECALCULATION) OR RECALCULATIONS) (4N) ((((((APPROVE) OR  
 APPROVES) OR APPROVED) OR APPROVER) OR APPROVERS) OR  
 APPROVAL) OR APPROVALS) OR APPROVING)  
 S8 1 S7 AND ((DYNAMIC OR DYNAMICALLY OR RECALCULATE OR  
 RECALCULATES OR RECALCULATED OR RECALCULATING OR  
 RECALCULATION OR RECALCULATIONS) (4N) (APPROVE OR APPROVES  
 OR APPROVED OR APPROVER OR APPROVERS OR APPROVAL OR  
 APPROVALS OR APPROVING))

? t s8/3/all

8/3/1 (Item 1 from file: 20)

DIALOG(R)File 20: Dialog Global Reporter

(c) 2011 Dialog. All rights reserved.

89726064 (USE FORMAT 7 OR 9 FOR FULLTEXT)

**Retailers Can Protect Margins and Ensure Smarter Spending With Coupa's New Retail Solution**

MARKETWIRE

August 03, 2011

**Journal Code:** MWIC **Language:** English **Record Type:** FULLTEXT

**Word Count:** 707

? t s8/3,k/all

8/3,K/1 (Item 1 from file: 20)

DIALOG(R)File 20: Dialog Global Reporter

(c) 2011 Dialog. All rights reserved.

89726064 (USE FORMAT 7 OR 9 FOR FULLTEXT)

**Retailers Can Protect Margins and Ensure Smarter Spending With Coupa's New Retail Solution**

MARKETWIRE

August 03, 2011

**Journal Code:** MWIC **Language:** English **Record Type:** FULLTEXT



```

Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processing
Processed 10 of 49 files ...
Processing
Processing
Processing
Processing
Processing
Processing
Processed 20 of 49 files ...
Processing
Processed 30 of 49 files ...
Processing
Processed 40 of 49 files ...
Processing
Completed processing all files
1470 S7
6988971 UPDATE
1855611 UPDATES
2139517 UPDATED
580572 UPDATING
94743404 NEW
577678 MODIFY
263941 MODIFYING
1380238 MODIFIED
59041 MODIFIES
263941 MODIFYING
1730870 APPROVE
629742 APPROVES
7731809 APPROVED
3162 APPROVER
2840 APPROVERS
5963781 APPROVAL
1162822 APPROVALS
411537 APPROVING
750756 (((((((((UPDATE OR UPDATES) OR UPDATED) OR UPDATING) OR
NEW) OR MODIFY) OR MODIFYING) OR MODIFIED) OR MODIFIES)
OR MODIFYING) (4N) (((((((APPROVE OR APPROVES) OR APPROVED)

```



OR APPROVER) OR APPROVERS) OR APPROVAL) OR APPROVALS) OR  
APPROVING)  
S9 44 S7 AND ((UPDATE OR UPDATES OR UPDATED OR UPDATING OR NEW  
OR MODIFY OR MODIFYING OR MODIFIED OR MODIFIES OR  
MODIFYING) (4N) (APPROVE OR APPROVES OR APPROVED OR  
APPROVER OR APPROVERS OR APPROVAL OR APPROVALS OR  
APPROVING))

? rd

>>>Record 702:4023994 incomplete bibliographic data - record retained in RD  
set

S10 29 RD (unique items)

? t s10/3/all

Dialog eLink: 

10/3/1 (Item 1 from file: 15)

DIALOG(R)File 15: ABI/Inform(R)

(c) 2011 ProQuest Info&Learning. All rights reserved.

07349571 2481131891

**Fraud: the profit killing disease**

Rich, Mike

NZ Business v25n9 pp: 42-43

Oct 2011

ISSN: 0113-4957 **Journal Code: NZBS**

**Word Count: 1235**

Dialog eLink: 

10/3/2 (Item 2 from file: 15)

DIALOG(R)File 15: ABI/Inform(R)

(c) 2011 ProQuest Info&Learning. All rights reserved.

06733889 2003362711

**Financial Supply Chain Review 2010**

Jack; Large, Wolfi

Euromoney pp: n/a

Mar 2010

ISSN: 0014-2433 **Journal Code: ERM**

**Word Count: 1919**

 Dialog eLink:

10/3/3 (Item 3 from file: 15)

DIALOG(R)File 15: ABI/Inform(R)

(c) 2011 ProQuest Info&Learning. All rights reserved.

02243018 84987630

**Managing with technology: automating budgeting from acquisitions**

Marilyn E. Barnes

Bottom Line v10n2 pp: 65-73

1997

**ISSN:** 0888-045X **Journal Code:** BTTL

**Word Count:** 3797

 Dialog eLink:

10/3/4 (Item 4 from file: 15)

DIALOG(R)File 15: ABI/Inform(R)

(c) 2011 ProQuest Info&Learning. All rights reserved.

01356303 00-07290

**Novell electronic commerce effort bogged down**

Burns, Christine

Network World v14n2 pp: 1, 12

Jan 13, 1997

**ISSN:** 0887-7661 **Journal Code:** NWW

**Word Count:** 560

 Dialog eLink:

10/3/5 (Item 5 from file: 15)

DIALOG(R)File 15: ABI/Inform(R)

(c) 2011 ProQuest Info&Learning. All rights reserved.

00749920 93-99141

**Too good to be true - How to avoid office-supply scams**

Maturi, Richard J

Office Systems v10n7 pp: 41-45

Jul 1993

**ISSN:** 8750-3441 **Journal Code:** OFS

**Word Count:** 1468

10/3/6 (Item 1 from file: 810)  
DIALOG(R)File 810: Business Wire  
(c) 1999 Business Wire . All rights reserved.

0670551 BW1267

**GREAT PLAINS SFTWR : Great Plains Software's Dynamics.NetApps Increase Efficiency of Business Processes**

February 10, 1997

**Byline:** Business Editors & Computer Writers

10/3/7 (Item 2 from file: 810)  
DIALOG(R)File 810: Business Wire  
(c) 1999 Business Wire . All rights reserved.

0588370 BW0185

**FLEXIINTERNATIONAL : FlexiInternational Software Announces FlexiWorkFlow; FlexiWorkFlow Enhances FlexiFinancials with Prebuilt Templates to Automate Key Business Processes**

May 22, 1996

**Byline:** Business/Technology Editors

10/3/8 (Item 1 from file: 275)  
DIALOG(R)File 275: Gale Group Computer DB(TM)  
(c) 2011 Gale/Cengage. All rights reserved.

01373667 **Supplier Number:** 09453537 (Use **Format 7 Or 9 For FULL TEXT** )  
**Cost-justifying computers. (column)**

Zachmann, William F.  
PC Magazine , v9 , n17 , p93(2)  
Oct 16 , 1990

**Document Type:** column

ISSN: 0888-8507

**Language:** ENGLISH **Record Type:** FULLTEXT; ABSTRACT

**Word Count:** 1140 **Line Count:** 00090

10/3/9 (Item 1 from file: 624)  
DIALOG(R)File 624: McGraw-Hill Publications

(c) 2011 McGraw-Hill Co. Inc. All rights reserved.



00817877

**Marketwatch**

Coal Week , Vol. 22, No. 48 , Pg 5

November 25, 1996

**Journal Code:** COW

**Section Heading:** Marketwatch **ISSN:** 0149-578X

**Word Count:** 1,853

10/3/10 (Item 1 from file: 621)

DIALOG(R)File 621: Gale Group New Prod.Annou.(R)

(c) 2011 Gale/Cengage. All rights reserved.

04126304 **Supplier Number:** 132140857 (USE FORMAT 7 FOR FULLTEXT)

**CIBER Concludes Oracle Financials Upgrade for Minneapolis Public Housing Authority.**

PR Newswire , p NA

Jan 16 , 2003

**Language:** English **Record Type:** Fulltext

**Document Type:** Newswire ; Trade

**Word Count:** 662

10/3/11 (Item 2 from file: 621)

DIALOG(R)File 621: Gale Group New Prod.Annou.(R)

(c) 2011 Gale/Cengage. All rights reserved.

04055039 **Supplier Number:** 131677507 (USE FORMAT 7 FOR FULLTEXT)

**Rush-Presbyterian to Expand OmniBuyer(R) Use; Omnicell Releases New Version Of  
OmniBuyer as Ascension Health Continues Rollout to 67 Acute Care Facilities In 20 States.**

PR Newswire , p NA

July 23 , 2003

**Language:** English **Record Type:** Fulltext

**Document Type:** Newswire ; Trade

**Word Count:** 819

10/3/12 (Item 3 from file: 621)

DIALOG(R)File 621: Gale Group New Prod.Annou.(R)

(c) 2011 Gale/Cengage. All rights reserved.

01790872 **Supplier Number:** 53587588 (USE FORMAT 7 FOR FULLTEXT)  
**Datastream Systems Announces Beta Release Of MP2(R) Messenger(TM).**

PR Newswire , p 6327  
Jan 19 , 1999

**Language:** English **Record Type:** Fulltext  
**Document Type:** Newswire ; Trade  
**Word Count:** 514

10/3/13 (Item 4 from file: 621)  
DIALOG(R)File 621: Gale Group New Prod.Annou.(R)  
(c) 2011 Gale/Cengage. All rights reserved.

01005434 **Supplier Number:** 39553892 (USE FORMAT 7 FOR FULLTEXT)  
**CINCOM ANNOUNCES PURCHASING SYSTEM ENHANCEMENTS**

PR Newswire , p N/A  
July , 1985

**Language:** English **Record Type:** Fulltext  
**Document Type:** Newswire ; Trade  
**Word Count:** 533

10/3/14 (Item 1 from file: 636)  
DIALOG(R)File 636: Gale Group Newsletter DB(TM)  
(c) 2011 Gale/Cengage. All rights reserved.

05300497 **Supplier Number:** 87749027 (USE FORMAT 7 FOR FULLTEXT)

**Extensity 6 launches in the UK; Extensity's ongoing technology leadership, new functionality, and improved reporting enhance integrated solution for managing employee-based financial processes.**

M2 Presswire , p 0  
June 25 , 2002

**Language:** English **Record Type:** Fulltext  
**Document Type:** Newswire ; Trade  
**Word Count:** 889

10/3/15 (Item 2 from file: 636)  
DIALOG(R)File 636: Gale Group Newsletter DB(TM)  
(c) 2011 Gale/Cengage. All rights reserved.

02172494 **Supplier Number:** 44085455 (USE FORMAT 7 FOR FULLTEXT)

**Delivery delays anger users**

Personal Computer Markets , p N/A

Sept 8 , 1993

**Language:** English **Record Type:** Fulltext

**Document Type:** Newsletter ; Trade

**Word Count:** 843

10/3/16 (Item 1 from file: 613)

DIALOG(R)File 613: PR Newswire

(c) 2011 PR Newswire Association Inc. All rights reserved.

01013806 20030723SFW039 (USE FORMAT 7 FOR FULLTEXT)

**Rush-Presbyterian to Expand OmniBuyer(R) Use**

PR Newswire

Wednesday , July 23, 2003 07:04 EDT

**Journal Code:** PR **Language:** ENGLISH **Record Type:** FULLTEXT **Document Type:** NEWSWIRE

**Word Count:** 776

10/3/17 (Item 2 from file: 613)

DIALOG(R)File 613: PR Newswire

(c) 2011 PR Newswire Association Inc. All rights reserved.

00920107 20030116LATH042 (USE FORMAT 7 FOR FULLTEXT)

**CIBER Concludes Oracle Financials Upgrade for Minneapolis**

PR Newswire

Thursday , January 16, 2003 12:02 EST

**Journal Code:** PR **Language:** ENGLISH **Record Type:** FULLTEXT **Document Type:** NEWSWIRE

**Word Count:** 620

10/3/18 (Item 3 from file: 613)

DIALOG(R)File 613: PR Newswire

(c) 2011 PR Newswire Association Inc. All rights reserved.

00785759 20020624SFM078 (USE FORMAT 7 FOR FULLTEXT)

**Extensity Announces General Availability of Extensity 6**

PR Newswire

Monday , June 24, 2002 07:52 EDT

**Journal Code:** PR **Language:** ENGLISH **Record Type:** FULLTEXT **Document Type:** NEWSWIRE

**Word Count:** 847

10/3/19 (Item 1 from file: 16)  
DIALOG(R)File 16: Gale Group PROMT(R)  
(c) 2011 Gale/Cengage. All rights reserved.

07542237 **Supplier Number:** 62950938 (USE FORMAT 7 FOR FULLTEXT)

**WebSphere vs. Domino.(IBM and Lotus' Internet /Web server software)(Product Information)**

Buchner, Mark  
MIDRANGE Systems , v 13 , n 6 , p 40  
May 1 , 2000  
**Language:** English **Record Type:** Fulltext  
**Document Type:** Tabloid ; Trade  
**Word Count:** 611

10/3/20 (Item 1 from file: 148)  
DIALOG(R)File 148: Gale Group Trade & Industry DB  
(c) 2011 Gale/Cengage. All rights reserved.

04133116 **Supplier Number:** 07804338 (USE FORMAT 7 OR 9 FOR FULL TEXT )  
**Making it in Taiwan: life at a Taipei computer company. (DTK Computer)**

Legg, Gary  
EDN , v34 , n21 , p57(6)  
Oct 12 , 1989  
ISSN: 0012-7515  
**Language:** ENGLISH  
**Record Type:** FULLTEXT; ABSTRACT  
**Word Count:** 2090 **Line Count:** 00160

10/3/21 (Item 2 from file: 148)  
DIALOG(R)File 148: Gale Group Trade & Industry DB  
(c) 2011 Gale/Cengage. All rights reserved.

02027864 **Supplier Number:** 03205733 (USE FORMAT 7 OR 9 FOR FULL TEXT )  
**Astronomers take inside track in San Diego lighting battle.**

Research & Development , v26 , p76(1)  
April , 1984  
**Language:** ENGLISH  
**Record Type:** FULLTEXT

**Word Count: 703    Line Count: 00053**

10/3/22 (Item 1 from file: 20)  
DIALOG(R)File 20: Dialog Global Reporter  
(c) 2011 Dialog. All rights reserved.

**87713938 (USE FORMAT 7 OR 9 FOR FULLTEXT)**  
**Coupa Sets New Standard for Ease of Use in Business Software**

MARKETWIRE  
May 18, 2011  
**Journal Code: MWIC    Language: English    Record Type: FULLTEXT**  
**Word Count: 659**

10/3/23 (Item 2 from file: 20)  
DIALOG(R)File 20: Dialog Global Reporter  
(c) 2011 Dialog. All rights reserved.

**87537076 (USE FORMAT 7 OR 9 FOR FULLTEXT)**  
**Changing of the guard**

Jo Ann Hustis  
MCCLATCHY-TRIBUNE REGIONAL NEWS - THE MODESTO BEE - CALIFORNIA  
May 11, 2011  
**Journal Code: KMOB    Language: English    Record Type: FULLTEXT**  
**Word Count: 614**

10/3/24 (Item 1 from file: 608)  
DIALOG(R)File 608: MCT Information Svc.  
(c) 2011 MCT Information Svc. All rights reserved.

**15685342 (USE FORMAT 7 OR 9 FOR FULLTEXT)**  
**Low turnout at LG town meeting**

Melissa McClendon  
Daily Times  
March 24, 2011  
**Document Type: NEWSPAPER    Record Type: FULLTEXT    Language: ENGLISH**  
**Word Count: 299**

10/3/25 (Item 2 from file: 608)  
DIALOG(R)File 608: MCT Information Svc.  
(c) 2011 MCT Information Svc. All rights reserved.



13319259 (USE FORMAT 7 OR 9 FOR FULLTEXT)

**Christmas Eve storm costs county \$253,259**

Karen Brady

Express-Star

January 12, 2010

**Document Type:** NEWSPAPER **Record Type:** FULLTEXT **Language:** ENGLISH

**Word Count:** 405

10/3/26 (Item 1 from file: 471)

DIALOG(R)File 471: New York Times Fulltext

(c) 2011 The New York Times. All rights reserved.

00659896 **NYT Sequence Number:** 239805830619 (USE FORMAT 7 FOR FULLTEXT)

**LONG ISLAND JOURNAL**

New York Times , Late City Final Edition ED , Col 1 , p 3

Sunday June 19 1983

**Document Type:** Newspaper **Language:** English

**Record Type:** Fulltext **Section Heading:** SECT11

**Word Count:** 1375

10/3/27 (Item 1 from file: 494)

DIALOG(R)File 494: St LouisPost-Dispatch

(c) 2011 St Louis Post-Dispatch. All rights reserved.

07256042

## **TIGHTER PURCHASING RULES PLANNED**

St. Louis Post Dispatch ( SL ) - MONDAY, September 13, 1993

**By:** Ralph Dummit

Of the St. Charles Post

**Edition:** FIVE STAR **Section:** ST. CHARLES **Page:** 08

**Word Count:** 503

10/3/28 (Item 1 from file: 702)

DIALOG(R)File 702: Miami Herald

(c) 2011 The Miami Herald Publishing Co. All rights reserved.

09715034


## **YOUR TOWN THIS WEEK**

Miami Herald ( MH ) - Monday, August 3, 1998  
**By:** Herald Staff  
**Edition:** Broward **Section:** Broward News **Page:** 2BR  
**Word Count:** 729

10/3/29 (Item 2 from file: 702)  
DIALOG(R)File 702: Miami Herald  
(c) 2011 The Miami Herald Publishing Co. All rights reserved.

04023994  
Miami Herald ( MH ) - SAT MAR 28 1987  
**By:** Herald Staff  
**Edition:** BRWRD **Section:** BRWD S **Page:** 2BR  
**Word Count:** 1,354

? t s10/3,k/4,7,11,13,14,17

**Dialog eLink:**   
10/3,K/4 (Item 4 from file: 15)  
DIALOG(R)File 15: ABI/Inform(R)  
(c) 2011 ProQuest Info&Learning. All rights reserved.

01356303 00-07290  
**Novell electronic commerce effort bogged down**

Burns, Christine  
Network World v14n2 pp: 1, 12  
Jan 13, 1997  
**ISSN:** 0887-7661 **Journal Code:** NWW  
**Word Count:** 560

**Text:**

...changes to their plan or provide a new ship date for the software. But they did confirm a beta release for late this quarter. The **new** plan is awaiting **approval** from company President Joseph Marengi.

"I wish that I could give you a good technical reason why the product has slipped, but I can't...

...server would encrypt transaction data and pass off transaction requests to a dedicated, secure server that could check credit, charge an account and relay an **approved purchase order** to a fulfillment house.

Nelson said Novell continues to work with Open Market, but he would not indicate whether the product- to be delivered later...

10/3,K/7 (Item 2 from file: 810)  
DIALOG(R)File 810: Business Wire  
(c) 1999 Business Wire . All rights reserved.

0588370 BW0185

**FLEXIINTERNATIONAL : FlexiInternational Software Announces FlexiWorkFlow;  
FlexiWorkFlow Enhances FlexiFinancials with Prebuilt Templates to Automate Key  
Business Processes**

May 22, 1996

**Byline:** Business/Technology Editors

...BUSINESS WIRE)--May 22, 1996--

FlexiInternational Software, Inc. (Flexi), a leading provider of open, object-oriented client/server financial management systems, today announced FlexiWorkFlow, its **new** suite of **approval** and notification workflow templates designed to automate the flow of information throughout an organization.

The FlexiWorkFlow templates provide organizations with ready-to-use solutions for important financial management processes, including: security; invoice and payment **approval**; requisitioning and **purchase order**; journal, account number and budget; credit collections, routing and write-offs; and asset and transfers. These prebuilt templates are designed as add-on modules to...

10/3,K/11 (Item 2 from file: 621)  
DIALOG(R)File 621: Gale Group New Prod.Annou.(R)  
(c) 2011 Gale/Cengage. All rights reserved.

04055039 **Supplier Number: 131677507 (USE FORMAT 7 FOR FULLTEXT)**  
**Rush-Presbyterian to Expand OmniBuyer(R) Use; Omnicell Releases New Version Of  
OmniBuyer as Ascension Health Continues Rollout to 67 Acute Care Facilities In 20 States.**

PR Newswire , p NA

July 23 , 2003

**Language:** English **Record Type:** Fulltext

**Document Type:** Newswire ; Trade

**Word Count:** 819

-

...engineering, and executive management, all before a purchase order is generated. In one case, a project manager was able to generate a requisition, get seven **approvals**, issue a **purchase order**, generate an invoice and get the vendor paid for an urgent

order all in six hours."  
Prevents Rogue Purchases  
OmniBuyer is a secure and fully...

...will benefit buyers and requisitioners. With the complexities and steps involved in the purchasing of such items, these rules will add significant efficiencies to the **approval** processes.

#### Additional Templates

**New** OmniBuyer features also include enhanced system administration functions and additional template management capability. Templates are now linked to the catalog update process, automatically cleaning up...

**20030723**

10/3,K/13 (Item 4 from file: 621)  
DIALOG(R)File 621: Gale Group New Prod.Annou.(R)  
(c) 2011 Gale/Cengage. All rights reserved.

01005434 **Supplier Number:** 39553892 (USE FORMAT 7 FOR FULLTEXT)  
**CINCOM ANNOUNCES PURCHASING SYSTEM ENHANCEMENTS**

PR Newswire, p N/A  
July, 1985

**Language:** English **Record Type:** Fulltext

**Document Type:** Newswire ; Trade

**Word Count:** 533

-  
...Cincom's  
CONTROL:Manufacturing family of products, comprising a "closed loop" manufacturing resource planning system.  
Major enhancements to the module include the management of "blanket" **purchase orders**, requisition **approval**  
, and general system enhancements.  
July, 1985

PURCHASING ENHANCEMENTS -Page 2  
BLANKET PURCHASE ORDER MANAGEMENT:  
\*The Purchasing module now gives organizations the ability to enter into...  
...the automatic maintenance of the blanket P.O. contract, or line item limits, based on the release quantity and extended line value.  
REQUISITION AND REQUISITION **APPROVAL** ENHANCEMENTS:  
\*A **new** requisition **approval** feature, designed specifically for items which are planned by MRP, is now included in the Purchasing module. Planned orders which are due to be released...

**19850701**

10/3,K/14 (Item 1 from file: 636)  
DIALOG(R)File 636: Gale Group Newsletter DB(TM)  
(c) 2011 Gale/Cengage. All rights reserved.

05300497 **Supplier Number:** 87749027 (USE FORMAT 7 FOR FULLTEXT)

**Extensity 6 launches in the UK; Extensity's ongoing technology leadership, new functionality, and improved reporting enhance integrated solution for managing employee-based financial processes.**

M2 Presswire , p 0

June 25 , 2002

**Language:** English **Record Type:** Fulltext

**Document Type:** Newswire ; Trade

**Word Count:** 889

-  
...not been submitted or credit card charges outstanding.  
-- The ability to take immediate action on outstanding items such as user policy administration and documents in **approval** queues.  
**New Functionality and Customer-Focused Enhancements** The depth of features in Extensity 6 reflects customer input and Extensity's distinct expertise in financial ERM processes. New worked, or specific project codes charged.  
-- Cheque request management to control and streamline vendor payments that are not associated with existing **purchase orders**.  
-- Streamlined **approval** processes including email approval and the ability to assign proxy reviewers for busy or out-of-the office managers.  
-- Improved support for Level 2 and...

**20020625**

10/3,K/17 (Item 2 from file: 613)  
DIALOG(R)File 613: PR Newswire  
(c) 2011 PR Newswire Association Inc. All rights reserved.

00920107 20030116LATH042 (USE FORMAT 7 FOR FULLTEXT)  
**CIBER Concludes Oracle Financials Upgrade for Minneapolis**

PR Newswire

Thursday , January 16, 2003 12:02 EST

**Journal Code:** PR **Language:** ENGLISH **Record Type:** FULLTEXT **Document Type:** NEWswire

**Word Count:** 620

**Text:**

...implementation of the Oracle Application Desktop Integrator and Workflow.  
Business process reengineering was provided to help MPHA integrate the use

of  
requisitions in initiating and **approving Purchase  
Orders**, and **new** features  
were implemented to support MPHA's year-end closing. The entire project  
was  
completed in just 13 weeks.  
"We were pleased to help MPHA...

? t s10/9/14

10/9/14 (Item 1 from file: 636)  
DIALOG(R)File 636: Gale Group Newsletter DB(TM)  
(c) 2011 Gale/Cengage. All rights reserved.

05300497 **Supplier Number: 87749027 (THIS IS THE FULLTEXT)**

**Extensity 6 launches in the UK; Extensity's ongoing technology leadership, new  
functionality, and improved reporting enhance integrated solution for managing employee-  
based financial processes.**

M2 Presswire, p 0  
June 25, 2002

**Language:** English **Record Type:** Fulltext

**Document Type:** Newswire ; Trade

**Word Count:** 889

**Text:**

RDATE:06262002

WEYBRIDGE -- Extensity (r), Inc. (Nasdaq: EXTN), a leading provider of Employee Relationship Management (ERM) solutions, today announced the launch of Extensity 6. Extensity 6 is a comprehensive solution that helps large organisations improve bottom line results through integrated tracking and controlling of employee-based financial processes for expense reporting, time capture, travel and procurement.

"In most corporations, up to 40 percent of all expenditures is in the hands of employees. Through better visibility into employee spending, organisations can make informed business decisions that favourably impact financial performance," said Bob Spinner, president and CEO of Extensity.

"With global deployments in more than 40 countries, Extensity has the expertise to help customers quickly identify cost savings and drive behaviours that positively impact the bottom line."

Extensity's unrivalled expertise in ERM processes are reflected in the new product features of Extensity 6, which include the following:

Rich, Real-Time Reporting and Analytics Extensity 6 delivers rich analytics and at-a-glance real-time monitoring across the corporation. Through an optimised reporting database and Extensity's architecture, Extensity 6 provides:

- Rich analytics through standard reports or a variety of customised reports using customers' own tools or Business Objects' tools that are embedded in Extensity 6.
- A dash-board, real-time view into time-sensitive data such as

time sheets that have not been submitted or credit card charges outstanding.

- The ability to take immediate action on outstanding items such as user policy administration and documents in **approval** queues.

**New Functionality and Customer-Focused Enhancements** The depth of features in Extensity 6 reflects customer input and Extensity's distinct expertise in financial ERM processes. New to Extensity 6 are the following features:

- A flexible Overtime Calculation Engine that can calculate overtime based on any combination, total hours per day, week, or month, special days worked, or specific project codes charged.

- Cheque request management to control and streamline vendor payments that are not associated with existing **purchase orders**.

- Streamlined **approval** processes including email approval and the ability to assign proxy reviewers for busy or out-of-the office managers.

- Improved support for Level 2 and Level 3 corporate and purchase card data with the ability to map all available card feed data to customer defined data fields.

- Enhancements to existing features including flexible barcode and document printing, and improved email notification management, to name a few.

Improved Usability and Easy Deployment Across the Enterprise  
Extensity 6 adds to Extensity's ease-of-use and ease-of-deployment across the enterprise. Extensity 6 provides:

- The first enterprise software available that leverages the power of Java Web Start, a new technology to access, update, and launch applications as easily as accessing a Web page.

- A more intuitive, easy-to-navigate interface as well as convenient, web-based access from almost any device.

- Automatic deployment to provide employee access anytime and anywhere, whether connected to the network or not.

- J2EE and improved scaling for large, geographically dispersed corporations that may have thousands of users who hit network at the same time, i.e. employees entering time sheets on a Friday afternoon.

Combined with Extensity's suite of services, Extensity Advantage, Extensity 6 gives customers a platform for employee-based financial processes that includes not only software, but services such as benchmarking, ROI analysis and expert deployment services, that enhance the customer's investment.

A.T. Kearney and Trimble Navigation are among the first of Extensity's customers worldwide to adopt Extensity 6.

"A single, easy to use system to manage employee spending is critical to achieve global efficiencies across a distributed workforce," said James Haddow, Director of Corporate Procurement for AT Kearney.

"Through new features such as pre-spending controls for travel and purchasing, Extensity 6 helps provide better cost management and improved insight into new areas of savings."

"With a geographically-dispersed employee base, usability and performance are critical. We're experiencing real benefit from our existing Extensity system and believe that the great leap forward in the analytics, deployability and user interface of Extensity 6 could further enhance our overall productivity," said John McCorquodale, Infrastructure Partner, Ernst & Young UK.

About Extensity

Extensity is a leading provider of Employee Relationship Management

(ERM) solutions. ERM helps enterprises control costs and improve employee effectiveness, enabling every employee to have a positive impact on the bottom line. The Extensity ERM suite automates employee-based financial processes, including travel planning and expense reporting, billable and payroll time capture and procurement, in a single, consolidated solution that provides efficiency, control, and effectiveness through analytics.

Extensity has licensed more than 1,000,000 seats worldwide to companies like Allianz, A.T. Kearney, Aventis, Cisco Systems, Merck, and Office Depot.

More information about Extensity is available on its Web site located at <http://www.extensity.com>.

CONTACT: Marcus Stanton/Katie Kinnes, CBC Tel: +44 (0)1483 277 711 e-mail: [marcuss@chazb.com](mailto:marcuss@chazb.com) e-mail: [katiek@chazb.com](mailto:katiek@chazb.com) Diane Paternoster, Extensity Tel: +44 (0)1932 268 720 e-mail: [dpaternoster@extensity.com](mailto:dpaternoster@extensity.com)

((M2 Communications Ltd disclaims all liability for information provided within M2 PressWIRE. Data prepared by named party/parties. Further information on M2 PressWIRE can be obtained at <http://www.presswire.net> on the world wide web. Inquiries to [info@m2.com](mailto:info@m2.com))).

THIS IS THE FULL TEXT: COPYRIGHT 2002 M2 Communications Ltd.  
COPYRIGHT 2002 Gale Group

**Publisher Name:** M2 Communications Ltd.

**Industry Names:** BUSN (Any type of business); INTL (Business, International )

? s1 and extensity and (approve or approves or approver or approvers or approval or approvals or approving or signoff or approved)

Processing

Processing

Processing

Processed 30 of 49 files ...

Completed processing all files

```
105294171 S1
3957 EXTENSITY
1730870 APPROVE
629742 APPROVES
3162 APPROVER
2840 APPROVERS
5963781 APPROVAL
1162822 APPROVALS
411537 APPROVING
7330 SIGNOFF
7731809 APPROVED
S11 583 S1 AND EXTENSITY AND (APPROVE OR APPROVES OR APPROVER OR
APPROVERS OR APPROVAL OR APPROVALS OR APPROVING OR
SIGNOFF OR APPROVED)
```

? s1 and extensity and ((approve or approves or approver or approvers or approval or approvals or approving or signoff or approved)(5n)(update or updates or updated or updating))



Processing  
Processing  
Processing  
Processing

Processed 10 of 49 files ...  
Completed processing all files

```

105294171 S1
    3957 EXTENSITY
    1730870 APPROVE
    629742 APPROVES
    3162 APPROVER
    2840 APPROVERS
    5963781 APPROVAL
    1162822 APPROVALS
    411537 APPROVING
    7330 SIGNOFF
    7731809 APPROVED
    6988971 UPDATE
    1855611 UPDATES
    2139517 UPDATED
    580572 UPDATING
    21103 (((((((APPROVE OR APPROVES) OR APPROVER) OR APPROVERS)
OR APPROVAL) OR APPROVALS) OR APPROVING) OR SIGNOFF) OR
APPROVED) (5N) (((UPDATE OR UPDATES) OR UPDATED) OR
UPDATING))
S12      0 S1 AND EXTENSITY AND ((APPROVE OR APPROVES OR APPROVER OR
APPROVERS OR APPROVAL OR APPROVALS OR APPROVING OR
SIGNOFF OR APPROVED) (5N) (UPDATE OR UPDATES OR UPDATED OR
UPDATING))

```

? ds

Set	Items	Description
S1	105294171	PD<20031017
S2	1	AU=(CIRULLI, S OR CIRULLI S? OR (SUSAN(2N)CIRULLI)) OR BY=-(SUSAN(2N)CIRULLI)
S3	4	AU=(HAGER, D OR HAGER D? OR ((DAN OR DANNY) (2N)HAGER)) OR -BY=((DAN OR DANNY) (2N)HAGER)
S4	5	S2 OR S3
S5	90	S1 AND ((DAN OR DANNY) (2N)HAGER) OR (SUSAN(2N)CIRULLI)
S6	0	S5 AND (APPROVE OR APPROVER OR APPROVED OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING) AND (PURCHASE(W) (FORM OR FORMS OR ORDER OR ORDERS OR REQUISITION OR REQUISITIONS))
S7	1470	S1 AND ((APPROVE OR APPROVER OR APPROVED OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING) (2N) (PURCHASE(W) (ORDER OR ORDERS OR REQUISITION OR REQUISITIONS)))
S8	1	S7 AND ((DYNAMIC OR DYNAMICALLY OR RECALCULATE OR RECALCULATES OR RECALCULATED OR RECALCULATING OR RECALCULATION OR RECALCULATIONS) (4N) (APPROVE OR APPROVES OR APPROVED OR APPROVER OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING))
S9	44	S7 AND ((UPDATE OR UPDATES OR UPDATED OR UPDATING OR NEW OR MODIFY OR MODIFYING OR MODIFIED OR MODIFIES OR MODIFYING) (4N) (APPROVE OR APPROVES OR APPROVED OR APPROVER OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING))
S10	29	RD (unique items)
S11	583	S1 AND EXTENSITY AND (APPROVE OR APPROVES OR APPROVER OR A-

PPROVERS OR APPROVAL OR APPROVALS OR APPROVING OR SIGNOFF OR -  
APPROVED)  
S12 0 S1 AND EXTENSITY AND ((APPROVE OR APPROVES OR APPROVER OR -  
APPROVERS OR APPROVAL OR APPROVALS OR APPROVING OR SIGNOFF OR  
APPROVED) (5N) (UPDATE OR UPDATES OR UPDATED OR UPDATING) )

? s s7 and extensity

1470 S7  
3957 EXTENSITY  
S13 6 S7 AND EXTENSITY

? t s13/3/all

13/3/1 (Item 1 from file: 610)  
DIALOG(R)File 610: Business Wire  
(c) 2011 Business Wire. All rights reserved.

00735057 20020624175B8636 (USE FORMAT 7 FOR FULLTEXT)  
**Elite Announces New Travel Expense Manager Powered by Extensity 6; Integrated  
Application Will Help Professional Services Firms Track and Control Costs**

Business Wire  
Monday , June 24, 2002 08:03 EDT  
**Journal Code: BW Language: ENGLISH Record Type: FULLTEXT Document Type:  
NEWSWIRE  
Word Count: 731**

13/3/2 (Item 1 from file: 275)  
DIALOG(R)File 275: Gale Group Computer DB(TM)  
(c) 2011 Gale/Cengage. All rights reserved.

02296365 **Supplier Number: 54624321 (Use Format 7 Or 9 For FULL TEXT )**  
**Cisco saves millions with staff intranet.(Company Operations)**

Goodwin, Bill  
Computer Weekly , 12(1)  
May 6 , 1999  
ISSN: 0010-4787  
**Language: English Record Type: Fulltext  
Word Count: 289 Line Count: 00027**

13/3/3 (Item 1 from file: 621)  
DIALOG(R)File 621: Gale Group New Prod.Annou.(R)  
(c) 2011 Gale/Cengage. All rights reserved.

03208546 **Supplier Number: 87694216 (USE FORMAT 7 FOR FULLTEXT)**

**Elite Announces New Travel Expense Manager Powered by Extensity 6; Integrated Application Will Help Professional Services Firms Track and Control Costs.**

Business Wire , p 0090

June 24 , 2002

**Language:** English **Record Type:** Fulltext

**Document Type:** Newswire ; Trade

**Word Count:** 776

13/3/4 (Item 1 from file: 636)

DIALOG(R)File 636: Gale Group Newsletter DB(TM)

(c) 2011 Gale/Cengage. All rights reserved.

05300497 **Supplier Number:** 87749027 (USE FORMAT 7 FOR FULLTEXT)

**Extensity 6 launches in the UK; Extensity's ongoing technology leadership, new functionality, and improved reporting enhance integrated solution for managing employee-based financial processes.**

M2 Presswire , p 0

June 25 , 2002

**Language:** English **Record Type:** Fulltext

**Document Type:** Newswire ; Trade

**Word Count:** 889

13/3/5 (Item 1 from file: 613)

DIALOG(R)File 613: PR Newswire

(c) 2011 PR Newswire Association Inc. All rights reserved.

00785759 20020624\$FM078 (USE FORMAT 7 FOR FULLTEXT)

**Extensity Announces General Availability of Extensity 6**

PR Newswire

Monday , June 24, 2002 07:52 EDT

**Journal Code:** PR **Language:** ENGLISH **Record Type:** FULLTEXT **Document Type:** NEWSWIRE

**Word Count:** 847

13/3/6 (Item 1 from file: 16)

DIALOG(R)File 16: Gale Group PROMT(R)

(c) 2011 Gale/Cengage. All rights reserved.

09885877 **Supplier Number:** 87694216 (USE FORMAT 7 FOR FULLTEXT)

**Elite Announces New Travel Expense Manager Powered by Extensity 6; Integrated**

# Application Will Help Professional Services Firms Track and Control Costs.

Business Wire , p 0090

June 24 , 2002

Language: English Record Type: Fulltext

Document Type: Newswire ; Trade

Word Count: 776

? ds

Set	Items	Description
S1	105294171	PD<20031017
S2	1	AU=(CIRULLI, S OR CIRULLI S? OR (SUSAN(2N)CIRULLI)) OR BY=(SUSAN(2N)CIRULLI)
S3	4	AU=(HAGER, D OR HAGER D? OR ((DAN OR DANNY) (2N)HAGER)) OR - BY=((DAN OR DANNY) (2N)HAGER)
S4	5	S2 OR S3
S5	90	S1 AND ((DAN OR DANNY) (2N)HAGER) OR (SUSAN(2N)CIRULLI)
S6	0	S5 AND (APPROVE OR APPROVER OR APPROVED OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING) AND (PURCHASE(W) (FORM OR FORMS OR ORDER OR ORDERS OR REQUISITION OR REQUISITIONS))
S7	1470	S1 AND ((APPROVE OR APPROVER OR APPROVED OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING) (2N) (PURCHASE(W) (ORDER OR ORDERS OR REQUISITION OR REQUISITIONS))))
S8	1	S7 AND ((DYNAMIC OR DYNAMICALLY OR RECALCULATE OR RECALCULATES OR RECALCULATED OR RECALCULATING OR RECALCULATION OR RECALCULATIONS) (4N) (APPROVE OR APPROVES OR APPROVED OR APPROVER - OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING))
S9	44	S7 AND ((UPDATE OR UPDATES OR UPDATED OR UPDATING OR NEW OR MODIFY OR MODIFYING OR MODIFIED OR MODIFIES OR MODIFYING) (4N - ) (APPROVE OR APPROVES OR APPROVED OR APPROVER OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING))
S10	29	RD (unique items)
S11	583	S1 AND EXTENSITY AND (APPROVE OR APPROVES OR APPROVER OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING OR SIGNOFF OR - APPROVED)
S12	0	S1 AND EXTENSITY AND ((APPROVE OR APPROVES OR APPROVER OR - APPROVERS OR APPROVAL OR APPROVALS OR APPROVING OR SIGNOFF OR APPROVED) (5N) (UPDATE OR UPDATES OR UPDATED OR UPDATING))
S13	6	S7 AND EXTENSITY

? b 347,348,349

20nov11 17:49:03 User264751 Session D859.2

\$13.01	2.219	DialUnits	File15
\$11.46	6	Type(s)	in Format 3
\$11.46	6	Types	
\$24.47	Estimated cost	File15	
\$5.96	0.997	DialUnits	File9
\$5.96	Estimated cost	File9	
\$1.01	0.907	DialUnits	File610
\$1.59	1	Type(s)	in Format 3
\$1.59	1	Types	
\$2.60	Estimated cost	File610	
\$0.54	0.486	DialUnits	File810

\$4.80 3 Type(s) in Format 3  
 \$4.80 3 Types  
 \$5.34 Estimated cost File810  
 \$4.48 0.750 DialUnits File275  
 \$0.00 2 Type(s) in Format 66  
 \$0.00 2 Types  
 \$4.48 Estimated cost File275  
 \$4.20 0.673 DialUnits File624  
 \$1.83 1 Type(s) in Format 3  
 \$1.83 1 Types  
 \$6.03 Estimated cost File624  
 \$10.57 1.768 DialUnits File621  
 \$15.75 7 Type(s) in Format 3  
 \$15.75 7 Types  
 \$26.32 Estimated cost File621  
 \$11.73 2.002 DialUnits File636  
 \$6.48 4 Type(s) in Format 3  
 \$4.10 1 Type(s) in Format 9  
 \$10.58 5 Types  
 \$22.31 Estimated cost File636  
 \$1.31 1.182 DialUnits File613  
 \$7.95 5 Type(s) in Format 3  
 \$7.95 5 Types  
 \$9.26 Estimated cost File613  
 \$0.76 0.684 DialUnits File813  
 \$0.76 Estimated cost File813  
 \$25.90 4.419 DialUnits File16  
 \$3.58 2 Type(s) in Format 3  
 \$3.58 2 Types  
 \$29.48 Estimated cost File16  
 \$4.43 0.756 DialUnits File160  
 \$4.43 Estimated cost File160  
 \$0.45 0.403 DialUnits File634  
 \$0.45 Estimated cost File634  
 \$34.78 5.817 DialUnits File148  
 \$3.86 2 Type(s) in Format 3  
 \$3.86 2 Types  
 \$38.64 Estimated cost File148  
 \$17.05 12.722 DialUnits File20  
 \$12.64 8 Type(s) in Format 3  
 \$12.64 8 Types  
 \$29.69 Estimated cost File20  
 \$0.73 0.164 DialUnits File35  
 \$0.00 1 Type(s) in Format 66  
 \$0.00 1 Types  
 \$0.73 Estimated cost File35  
 \$3.06 0.822 DialUnits File583  
 \$3.06 Estimated cost File583  
 \$0.46 0.100 DialUnits File65  
 \$0.46 Estimated cost File65  
 \$29.18 2.202 DialUnits File2  
 \$29.18 Estimated cost File2  
 \$3.41 0.875 DialUnits File474  
 \$3.41 Estimated cost File474  
 \$1.33 0.341 DialUnits File475  
 \$1.33 Estimated cost File475  
 \$2.47 0.473 DialUnits File99

\$2.47 Estimated cost File99  
       \$0.40 0.070 DialUnits File256  
 \$0.40 Estimated cost File256  
       \$6.10 1.020 DialUnits File635  
 \$6.10 Estimated cost File635  
       \$3.85 0.635 DialUnits File570  
 \$3.85 Estimated cost File570  
       \$7.13 1.193 DialUnits File47  
 \$7.13 Estimated cost File47  
       \$3.08 2.771 DialUnits File608  
       \$3.18 2 Type(s) in Format 3  
       \$3.18 2 Types  
 \$6.26 Estimated cost File608  
       \$0.22 0.202 DialUnits File387  
 \$0.22 Estimated cost File387  
       \$1.27 1.144 DialUnits File471  
       \$1.59 1 Type(s) in Format 3  
       \$1.59 1 Types  
 \$2.86 Estimated cost File471  
       \$0.61 0.552 DialUnits File492  
 \$0.61 Estimated cost File492  
       \$0.58 0.522 DialUnits File494  
       \$1.59 1 Type(s) in Format 3  
       \$1.59 1 Types  
 \$2.17 Estimated cost File494  
       \$0.71 0.643 DialUnits File631  
 \$0.71 Estimated cost File631  
       \$0.46 0.417 DialUnits File633  
 \$0.46 Estimated cost File633  
       \$0.52 0.466 DialUnits File638  
 \$0.52 Estimated cost File638  
       \$0.33 0.294 DialUnits File640  
 \$0.33 Estimated cost File640  
       \$0.38 0.339 DialUnits File641  
 \$0.38 Estimated cost File641  
       \$0.74 0.662 DialUnits File702  
       \$3.18 2 Type(s) in Format 3  
       \$3.18 2 Types  
 \$3.92 Estimated cost File702  
       \$0.28 0.253 DialUnits File703  
 \$0.28 Estimated cost File703  
       \$0.55 0.492 DialUnits File704  
 \$0.55 Estimated cost File704  
       \$0.46 0.415 DialUnits File713  
 \$0.46 Estimated cost File713  
       \$0.39 0.349 DialUnits File714  
 \$0.39 Estimated cost File714  
       \$0.16 0.141 DialUnits File715  
 \$0.16 Estimated cost File715  
       \$0.13 0.115 DialUnits File725  
 \$0.13 Estimated cost File725  
       \$0.44 0.394 DialUnits File735  
 \$0.44 Estimated cost File735  
       \$0.25 0.226 DialUnits File477  
 \$0.25 Estimated cost File477  
       \$0.75 0.675 DialUnits File710  
 \$0.75 Estimated cost File710

```

$0.36      0.328 DialUnits File711
$0.36 Estimated cost File711
$0.23      0.209 DialUnits File756
$0.23 Estimated cost File756
$0.79      0.709 DialUnits File757
$0.79 Estimated cost File757
OneSearch, 49 files, 55.998 DialUnits FileOS
$19.22 INTERNET
$310.79 Estimated cost this search
$311.01 Estimated total session cost 56.247 DialUnits

```

```

SYSTEM:OS - DIALOG OneSearch
File 347:JAPIO Dec 1976-2011/JULY(Updated 111028)
(c) 2011 JPO & JAPIO
File 348:EUROPEAN PATENTS 1978-201146
(c) 2011 European Patent Office
File 349:PCT FULLTEXT 1979-2011/UB=201111110|UT=20111103
(c) 2011 WIPO/Thomson

```

```

Set      Items      Description
---      -

```

? s pd<20021017

```

Processing
Processing
Processing
Processing
Processing
Processing
Processing

```

```

S1 9917694 PD<20021017

```

? s AU=(cirulli, s or cirulli s? or (susan(2n)cirulli) or BY=(susan(2n)cirulli)

```

>>>One or more prefixes are unsupported
>>> or undefined in one or more files.
      0 AU=CIRULLI, S
      0 AU=CIRULLI S?
    9956 AU=SUSAN
      8 AU=CIRULLI
      0 AU=SUSAN(2N)AU=CIRULLI
      0 BY=SUSAN
      0 BY=CIRULLI
      0 BY=SUSAN(2N)BY=CIRULLI
    S2      0 AU=(CIRULLI, S OR CIRULLI S? OR (SUSAN(2N)CIRULLI)) OR
      BY=(SUSAN(2N)CIRULLI)

```

? s AU=(hager, d or hager d? or ((dan or danny)(2n)hager)) or BY=((dan or danny)(2n)hager)

```

>>>One or more prefixes are unsupported
>>> or undefined in one or more files.

```

```

0 AU=HAGER, D
28 AU=HAGER D?
10994 AU=DAN
3169 AU=DANNY
918 AU=HAGER
0 (AU=DAN OR AU=DANNY) (2N)AU=HAGER
0 BY=DAN
0 BY=DANNY
0 BY=HAGER
0 (BY=DAN OR BY=DANNY) (2N)BY=HAGER
S3 28 AU=(HAGER, D OR HAGER D? OR ((DAN OR DANNY) (2N)HAGER)) OR
BY=((DAN OR DANNY) (2N)HAGER)

```

**? s s1 and (s2 or s3) and ((approve or approver or approvers or approval or approvals or approving or signoff) and (purchase(w)(form or forms or order or orders or requisition or requisitions))**

>>>Unmatched parentheses

**? s s1 and (s2 or s3) and (approve or approver or approvers or approval or approvals or approving or signoff) and (purchase(w)(form or forms or order or orders or requisition or requisitions))**

Processing

Processing

Processing

Processing

```

9917694 S1
0 S2
28 S3
7019 APPROVE
658 APPROVER
211 APPROVERS
33891 APPROVAL
2001 APPROVALS
3694 APPROVING
128 SIGNOFF
75501 PURCHASE
3339158 FORM
1239612 FORMS
2275294 ORDER
90186 ORDERS
2 REQUISITION
0 REQUISITIONS
3812 PURCHASE(W) (((((FORM OR FORMS) OR ORDER) OR ORDERS) OR
REQUISITION) OR REQUISITIONS)
S4 0 S1 AND (S2 OR S3) AND (APPROVE OR APPROVER OR APPROVERS
OR APPROVAL OR APPROVALS OR APPROVING OR SIGNOFF) AND
(PURCHASE(W)(FORM OR FORMS OR ORDER OR ORDERS OR
REQUISITION OR REQUISITIONS))

```

**? s s1 and (((DAN OR DANNY)(2N)HAGER) or (SUSAN(2N)CIRULLI))**

```

9917694 S1

```



```

23611 DAN
3099 DANNY
2007 HAGER
      0 (DAN OR DANNY) (2N) HAGER
10470 SUSAN
      62 CIRULLI
      0 SUSAN(2N)CIRULLI
S5      0 S1 AND (((DAN OR DANNY) (2N) HAGER) OR (SUSAN(2N)CIRULLI))

```

? s pd<20031017

Processing  
Processing  
Processing  
Processing  
Processing  
Processing  
Processing

S610391056 PD<20031017

? s s1 and (approve or approver or approvers or approval or approvals or approving or signoff) and (purchase(w)(form or forms or order or orders or requisition or requisitions))

Processing  
Processing  
Processing  
Processing

```

9917694 S1
      7019 APPROVE
      658 APPROVER
      211 APPROVERS
33891 APPROVAL
      2001 APPROVALS
      3694 APPROVING
      128 SIGNOFF
      75501 PURCHASE
3339158 FORM
1239612 FORMS
2275294 ORDER
      90186 ORDERS
      2 REQUISITION
      0 REQUISITIONS
      3812 PURCHASE(W) (((((FORM OR FORMS) OR ORDER) OR ORDERS) OR
      REQUISITION) OR REQUISITIONS)
S7      517 S1 AND (APPROVE OR APPROVER OR APPROVERS OR APPROVAL OR
      APPROVALS OR APPROVING OR SIGNOFF) AND (PURCHASE(W) (FORM
      OR FORMS OR ORDER OR ORDERS OR REQUISITION OR
      REQUISITIONS))

```

? s s7 and ((dynamic or dynamically or recalculate or recalculates or recalculated or recalculating or recalculation or recalculations)(4n)(approve or approves or approved or approver or approvers or approval or approvals or approving))

517 S7  
 329270 DYNAMIC  
 118056 DYNAMICALLY  
 4517 RECALCULATE  
 2814 RECALCULATES  
 10193 RECALCULATED  
 3513 RECALCULATING  
 3506 RECALCULATION  
 315 RECALCULATIONS  
 7019 APPROVE  
 4127 APPROVES  
 79727 APPROVED  
 658 APPROVER  
 211 APPROVERS  
 33891 APPROVAL  
 2001 APPROVALS  
 3694 APPROVING  
 108 (((((((DYNAMIC OR DYNAMICALLY) OR RECALCULATE) OR  
 RECALCULATES) OR RECALCULATED) OR RECALCULATING) OR  
 RECALCULATION) OR RECALCULATIONS) (4N) (((((((APPROVE OR  
 APPROVES) OR APPROVED) OR APPROVER) OR APPROVERS) OR  
 APPROVAL) OR APPROVALS) OR APPROVING)  
 S8 14 S7 AND ((DYNAMIC OR DYNAMICALLY OR RECALCULATE OR  
 RECALCULATES OR RECALCULATED OR RECALCULATING OR  
 RECALCULATION OR RECALCULATIONS) (4N) (APPROVE OR APPROVES  
 OR APPROVED OR APPROVER OR APPROVERS OR APPROVAL OR  
 APPROVALS OR APPROVING))

? t s8/3,k/all

DIALOG(R)File 348: EUROPEAN PATENTS  
 (c) 2011 European Patent Office. All rights reserved.  
 8/3K/1 (Item 1 from file: 348)  
 03658492

### Method and system to effectuate recovery for dynamic workflows

Verfahren und System zum Bewirken einer Wiederherstellung für dynamische Workflows  
 Procédé et système permettant d'effectuer la récupération de flux de travail dynamiques

#### Patent Assignee:

- SAP AG** (100981927)  
 Dietmar-Hopp-Allee 16; 69190 Walldorf (DE)  
 (Applicant designated States: all)

#### Inventor:

- Lu, Ruopeng**  
 c/o SAP AGGlobal IP GroupDietmar-Hopp-Allee 16; 69190 Walldorf; (DE)

- **Kowalkiewicz, Marek**  
c/o SAP AGGlobal IP GroupDietmar-Hopp-Allee 16; 69190 Walldorf; (DE)
- **Schaeufele, Bernd**  
c/o SAP AGGlobal IP GroupDietmar-Hopp-Allee 16; 69190 Walldorf; (DE)
- **Kruempelmann, Marita**  
c/o SAP AGGlobal IP GroupDietmar-Hopp-Allee 16; 69190 Walldorf; (DE)
- **Baeuerle, Stefan**  
c/o SAP AGGlobal IP GroupDietmar-Hopp-Allee 16; 69190 Walldorf; (DE)

#### Legal Representative:

- **Muller-Bore & Partner Patentanwalte et al (101215276)**  
Grafinger Strasse 2; 81671 Munchen; (DE)

	Country	Number	Kind	Date	
Patent	EP	2372620	A1	20111005	(Basic)
Application	EP	11001801		20110303	
Priorities	US	751553		20100331	

#### Designated States:

AL; AT; BE; BG; CH; CY; CZ; DE; DK; EE;  
ES; FI; FR; GB; GR; HR; HU; IE; IS; IT;  
LI; LT; LU; LV; MC; MK; MT; NL; NO; PL;  
PT; RO; RS; SE; SI; SK; SM; TR

#### Extended Designated States:

BA; ME

International Classification (Version 8) IPC	Level	Value	Position	Status	Version	Action	Source	Office
G06Q-0010/00	A	I	F	B	20060101	20110616	H	EP

**Abstract Word Count:** 72

**NOTE: Figure number on first page:** 1

**Language** Publication: English

Procedural: English

Application: English

Fulltext Availability	Available Text	Language	Update	Word Count
CLAIMS A		(English)	201140	1046
SPEC A		(English)	201140	5745
Total Word Count (Document A) 6791				

Fulltext Availability	Available Text	Language	Update	Word Count
Total Word Count (Document B) 0				
Total Word Count (All Documents) 6791				

**Specification:** ...new input data.

In large organizations, many tasks, such as those related to expenses, need to be approved before they are performed. For instance, a **purchase order** may need to be approved by several roles, depending on the value and the type of the items listed in the **purchase order**, and on the ordering agent's position in the organization's hierarchy. A purchase **approval** process may be expressed by a workflow model (e.g., titled a purchase workflow model). A computerized purchase **approval** process based on the purchase workflow model may be termed a purchase **approval** workflow. Each time a new **purchase order** needs to be processed, a new instance of the purchase **approval** workflow may be started.

As a purchase **approval** process may be more or less complex, the nature of business may require changes to a **purchase order** before an instance of the purchase **approval** workflow has completed execution. Currently it is not a common practice to enable changes to application data during workflow execution. The typical solution offered in... ..be evaluated before it can be determined whether the task should be re-executed.

Smart recovery process may be illustrated using an example of a **dynamic** purchase **approval** process. A diagrammatic representation of an **approval** workflow is illustrated in <FIGREF IDREF=F0001>Fig. 1</FIGREF>. The **approval** workflow is triggered when a project member creates a purchase request that involves a data object "**Purchase Order**" that is associated with a value that reflects the price of the items in the **Purchase Order**. Information present associated with the **Purchase Order** is provided in a form of a purchase request "PO" shown in <FIGREF IDREF=F0001>Fig.1</FIGREF> (block 102). According to the instance-specific conditions, (e.g., the price reflected in the purchase request), the purchase request is to be processed by several **approval** tasks, including "Cost Centre **Approval**" (block 108), "IT Manager **Approval**" (block 110), "Facility Manager **Approval**" (block 112), and "Dangerous Good **Approval**" (block 114). The value associated with the virtual **Purchase Order** can be changed at any time due to various reasons, and at some arbitrary time during the course of the workflow execution. For example, the... ..item cost from catalogue has to propagate to the current purchase requests.

When a change to the "PO" data object (block 102) occurs while the **approval** workflow is executing, the following basic questions may arise during the abovementioned changes on the workflow instance. First, it is determined whether the change to... ..workflow during runtime. Referring to the example illustrated in <FIGREF IDREF=F0001>Fig. 1</FIGREF>, it may be necessary to re-execute the "Dangerous Goods **Approval**" task whenever there is a change to the data item associated with the "**Purchase Order**." The task "Email Process Information," on the other hand, should be executed only once if it is not necessary to resend the email associated with the **purchase order**. Some tasks in the **approval** workflow require more complex execution behavior. For instance, the "Cost Center **Approval**" task may be considered highly important, but also resource-intensive, because it may need to be executed by a supervising employee.

Therefore, the "Cost Center **Approval**" task should not be re-executed every time, when an item of the **purchase order** is changed. It may be determined by the user who is designing the **approval** workflow model that this task should only be re-executed if certain conditions are met, e.g., if the value of the items in the **Purchase Order** has changed by a certain percentage.

As the re-evaluation labels are considered during re-execution at runtime, improperly assigned labels may lead to data... ..module may be implemented as a plug-in for an existing workflow editor. <FIGREF IDREF=F0004>Fig. 4</FIGREF> is a diagrammatic representation of the **dynamic approval** process shown in <FIGREF IDREF=F0001>Fig. 1</FIGREF>, where the tasks have been assigned re-evaluation labels. Also shown in <FIGREF IDREF=F0004>Fig. 4</FIGREF> is a pop-up warning window 402 presented in response to the validation module detecting potential redundant read by the task "IT Manager **Approval**."

Returning to <FIGREF IDREF=F0003>Fig. 3</FIGREF>, the auto completion function may be configured to assign appropriate labels to those tasks in a workflow...

DIALOG(R)File 348: EUROPEAN PATENTS

(c) 2011 European Patent Office. All rights reserved.

8/3K/2 (Item 2 from file: 348)

00836626

### **Method and apparatus for distributing conditional work flow processes among a plurality of users**

Verfahren und Vorrichtung zum Verteilen von konditionellen Arbeitsflussprozessen zwischen mehreren Benutzern

Methode et appareil pour la distribution de processus conditionnel de flux de travail entre plusieurs utilisateurs

#### **Patent Assignee:**

- **Dun & Bradstreet Software Services, Inc.** (2047260)  
3445 Peachtree Street, NE; Atlanta, Georgia 30326-1276 (US)  
(applicant designated states:  
AT;BE;CH;DE;DK;ES;FI;FR;GB;GR;IE;IT;LI;LU;MC;NL;PT;SE)

#### **Inventor:**

- **Rossi, Charles**  
1 Indian Meadow Drive; Northborough, Massachusetts 01532; (US)
- **Vinter, Stephen**  
23 Hundred Oaks Lane; Ashland, Massachusetts 01721; (US)

- **Conte, Leonard**  
281 Green Street; Northborough, Massachusetts 01532; (US)
- **Chang, S. Jay**  
3 Duggan Road; Acton, Massachusetts 01720; (US)
- **Botzer, Robert**  
49 Delmar Avenue; Framingham, Massachusetts 01701; (US)
- **McAllister, Sandra**  
28 Lee Street; Lancaster, Massachusetts 01523; (US)
- **Dorden, Joanne**  
2 Nottingham Road; Grafton, Massachusetts 01519; (US)

#### Legal Representative:

- **Brunner, Michael John (28871)**  
GILL JENNINGS & EVERY Broadgate House 7 Eldon Street; London EC2M 7LH; (GB)

	Country	Number	Kind	Date	
Patent	EP	774725	A2	19970521	(Basic)
Patent	EP	774725	A3	19981028	
Application	EP	96304925		19960703	
Priorities	US	557531		19951114	

#### Designated States:

AT; BE; CH; DE; DK; ES; FI; FR; GB; GR;  
IE; IT; LI; LU; MC; NL; PT; SE

**International Patent Class (V7):** G06F-017/60; ;

**Abstract** ...used to determine what the next step in the workflow should be, to determine who the next step should be assigned to, to select which **approvers** on an **approval** list are used, etc. Various types of conditional comparisons may be made in order to perform this functionality. Yet another feature of the present invention...

**Abstract Word Count:** 115

**Language** Publication: English

Procedural: English

Application: English

Fulltext Availability	Available Text	Language	Update	Word Count
CLAIMS A		(English)	EPAB97	946
SPEC A		(English)	EPAB97	41563
Total Word Count (Document A) 42509				

Fulltext Availability	Available Text	Language	Update	Word Count
Total Word Count (Document B) 0				
Total Word Count (All Documents) 42509				

**Specification:** ...work flow model.

An example of such a work flow is the routing of a document within an organization for the purpose of obtaining an **approval**. In a computer system for implementing a **purchase order** flow, a first user, such as a purchasing agent, initially creates a **purchase order** document. The creation of the **purchase order** document may include adding **purchase order** information to a computerized **purchase order** document image, drafting a **purchase order** computer document or a combination of both. After the **purchase order** document is created and at the command of the purchasing agent, the document may then be electronically routed to the second user, such as the purchasing manager, who could simply **approve**/disapprove the document or may add, delete or change information in the document prior to giving his/her **approval**. Finally, the document may be electronically routed to a third user, such as a finance or engineering manager, who may either **approve** or disapprove the **purchase order** document. The document may then be electronically routed back to the purchasing agent, who may then act upon the document accordingly.

A drawback associated with... ..the quantity and cost of the part. Nonetheless, both the original paper order system and its computerized implementation provide all of the information in the **purchase order** to everyone, rather than only the information relevant to each individual's specific needs. Where large documents are involved, each individual may be then forced to...used to determine what the next step in the workflow should be, to determine who the next step should be assigned to, to select which **approvers** on an **approval** list are used, etc. Various types of conditional comparisons may be made in order to perform this functionality.

Yet another feature of the present invention... activities/tasks 250 represented as messages 750 available to this user in his or her "New To Do List" To Do List window 700 are " **Approve** class registrations", "Registration confirmation," and "Select payment type for ...next activities/tasks 250 to the left of each message 750. In this example, the To Do list window 700, to the left of the "**Approve** class registrations" message 750, reveals that there are eight next activities/tasks 250 for this category, where six are new 770 and two are done... ..undertaken in this flow process. For this example, the next steps 230 are "Review Part Planning information" to be done by the manufacturing manager and "**approve** part planning" (not shown) to be done by the quality department manager.

The "review part planning information" message 750, representative of a next activity/task...alphabetical order where no sequence specific information for an activity is indicated.

For this example, the user has selected the Class Registration, Class Payment, Registration **Approval**, and Activity activities 1760 for his or her "Sample Class Registration" list 1750. Moreover, the user has chosen to sequentially list the activities such that the Class Registration

Activity has the highest SEQ(underscore)NBR, with Class Payment and Registration **Approval** having lower SEQ(underscore)NBRs and Activity having the lowest or no SEQ(underscore)NBR.

From the "Sample Class Registration" list window 1750, the user...see FIG. 12). Other next steps selected by the administrator include the "Class payment type selected" (pamsam2up1) event with the next activity/task being "Registration **Approval**" (pam0520) which is assigned to user RSD.

The administrator may also activate the archiving feature of the computer system of the present invention. This feature...230 of FIG. 2), to determine to whom the next step should be assigned (e.g., element 240 of FIG. 2), or to select which **approvers** on an **approval** list should be used. In one embodiment, the following operations may be used as conditional logic:

- \* Comparing an integer or numeric column to a constant... element 220 of FIG. 2) within conditional logic. An enhanced trigger event function may be implemented to allow applications to pass more than 16 values.

**Approval Lists.** Adding **approval** lists to the present invention adds the following functionality:

- \* Provide **approve** and reject as activity actions. This allows the **approver** to see the object that they are **approving**. Otherwise, **approvers** in a work flow use an **approval** window and have to zoom to the actual object.

- \* Provide a consistent handling of **approvals**. This ensures that all the **approval** windows have the same look and feel. It also decreases the development cost of adding **approvals** to new windows and the maintenance cost of the **approval** maintenance windows.

- \* Automatic support for new **Approval** features as they are added to the platform. These include substitutes, roles, and conditional generation.

- \* Allow users to **approve** items directly from the Task Details list (element 5601 of FIG. 56) without actually entering the application activity.

**Structures Integration.** The present invention may be...replace the previous workflow workbench with a new user interface that is more intuitive and which supports specification of conditional next steps, conditional assignments, and **approvals**.

**Mail Integration.** The present invention may be designed to support mail integration by creating an "attachment". Mail would be used as the delivery mechanism for...Data 10010, Workflow Definition 10015, and Message Queue tables 10050. The Workflow Engine 10045 may also include logic to resolve roles and substitutes, defined previously.

**Approvals** may be built into the workflow. **Approvals** build on the application specific **approval** processes that are already in place, and require the application developer to do the following:



1) Add an **approval** table in the application database. The key of **approval** table is the application table key, a sequence number, and an owner. The entries with the key matching the key of the application table row would be the **approval** list for that row.

2) The platform provides **approval** lists. When an **approval** is the next step the **approval** list to be used is passed to the **approval** window as data. Conditional logic can be supported in the determining of which **approval** list is to be used. **Dynamic** generation of the **approval** list will allow the name of the **approval** list to be used to be a field on the application window.

3) The platform will also provide a generalized window for tracking **approvals**. This window will be the management interface into the application provided **approval** table. The **approval** tracking window may either be an ancestor window that the applications use to create their own descendant tracking window with the proper keys or may...id or workgroup id of the substitute user or workgroup.

\* subst(underscore)owner(underscore)type - Indicates whether subst(underscore)owner is a user or workgroup. **approval**(underscore)list. The **approval** list table contains the header information for each **approval** list. It contains the following columns:

\* apprvl(underscore)list(underscore)id - The name of the **approval** list.

\* apprvl(underscore)activity(underscore)id - The activity the **approval** list is defined for.

\* notes - Notes that the user can use to describe an **approval** list. **approval**(underscore)list(underscore)detail. The **approval** list detail table contains a line item for each **approver** on the list. It contains the following columns:

\* apprvl(underscore)list(underscore)id - The name of the **approval** list.

\* apprvl(underscore)activity(underscore)id - The activity the **approval** list is defined for.

\* apprvl(underscore)level - The level for this **approver** on the list.

\* apprvl(underscore)id - The user ID, workgroup or role of the **approver**.

\* apprvl(underscore)type(underscore)code - The type of apprvl(underscore)id, either user (u), workgroup (g), role (r), or structure function (f).

\* structure(underscore)function...workflow engine will have to resolve the role.

Substitute processing - If the user has a substitute defined assign change the assignee to the substitute user.

**Approvals** -- **Approvals** are described in further detail in a later section. The new algorithm for the psp(underscore)trigger(underscore)ams(underscore)event stored procedure is described...  
...tables and return a boolean TRUE or FALSE to indicate the value of the expression.

Conditional workflow will preferably require the following application architecture changes:

**Approve, Reject.** **Approve** and **Reject** are few actions that are only available for windows that support **approvals**. **Approve** and **Reject** will execute the logic that is currently in the **approval** windows. These actions are described in further detail in a later section.

Fix step completion bugs. The logic within basic window which controls when a... ..postfix notation before storing them in the appropriate conditional table. This work flow workbench mapping tool is defined in further detail in a later section.

**Approval Tracking.** This window may be keyed by the application key. It may display the current status of the **approval** process for the item shown. It also shows a list of the future **approvers**. **Approvals** are described in further detail in a later section.

**Approval List Definition.** This window allows the administrator to create an **approval** list. Each list is keyed by the list name and the activity class for which it is applicable. The global activity class (\*) means that the **approval** list can be used for any window in the system that supports **approvals**. **Approvals** are described in further detail in a later section.

**Role Definition.** The Role Definition window is a simple tabular window used to maintain the role...structure functions as the assign to in next step options requires that structures be enhanced to include work flow assignee fields within the point definition.

## **Approvals**

### Definitions

**Approval Enabled Activity** - A window that supports user configuration of **Approval** workflow by calling the new **Approval** trig(underscore)event function and allowing for the possibility that no **Approval** process was required. The activity is identified to the workflow mapping tool through the new apprvl(underscore)enabled(underscore)activity table.

**Approval Enabled Event** - An event that allows users to map a workflow using an **Approval** List for distribution of Next Step messages.

**Approval List** - A list of Users, Workgroups, Roles or structures functions which supports sequential grouping used in an **Approval** process for an **Approval** Enabled Activity.

**Approval Tracking/Status** - An application window/table that supports tracking of a single instance of the **Approval** process and access to **Approval** comments. Shows **Approval** level, **approver**, name and type, **Approval** Status.

**Substitutes** - A User or Workgroup that will temporarily receive Workflow messages intended for a specified User. This will apply to ALL Workflow not just **Approvals**.

Workflow Mapping Tool - A completely new graphical user interface for defining and describing workflows. It is intuitive and supports specification of conditional Next Steps, conditional assignments and assignment of **Approval** Lists. It is described in further detail below.

The main goal of the **Approvals** portion of the present invention is to provide the user 120 with a way to customize the **Approval** process with the least amount of work required by the applications. A secondary goal is to provide a model, with supporting tools, to standardize the way we do **Approvals**. This will allow automatic support of new **Approval** features as they are added to the platform. These include Substitutes, Roles, and conditional generation

Four application windows which use an **Approval** process have been used as the basis for the design. Currently, each transaction window that supports **approvals** has:

- 1) An **Approval** List table, stored procedures and a maintenance window with a "Copy From" response window.
- 2) An **Approval** Substitutes table, stored procedures and a maintenance window.
- 3) An **Approval** Tracking/Status table to monitor the **approval** process of a specific instance, a comments table, stored procedures and a maintenance window.
- 4) Some mechanism for either **approving** or rejecting and handling the updates to tracking/status tables.
- 5) Scripts to read the instance table, generate the appropriate To Do messages and to recognize when the **Approval** process is complete.

Payment Request and Journal Entry **Approvals** are very similar in that they both use the common **Approval** List/Copy From and Substitutes ancestor windows and have an **Approval** Status table with associated Comments. Purchase Requisitions use a similar format for Lists and Substitutes but have these tables in a different table family than the Status and Comments tables. They also support **Approval** at both the line and the document level. All three use the **Approval** Tracking window as the Next Step activity in the **Approval** process, thus making the object being approved only visible at **Approval** time through Zoom. Requisitions provides an **Approval** view on the Requisition maintenance window. ECR/ECN seems the most different from the others with Comments associated with the document being approved rather than the Status table, **Approval** from the document window and different column names and datatypes for Lists and Substitutes.

To provide **Approval** Enabled activities, i.e. activities which support user modification of the **Approval** process through the use of the Workflow Mapping Tool, and to integrate new workflow features in to that process, the following tasks must be accomplished:

1.) Provide a mechanism to identify **Approval** Enabled activities and events and their components, e.g. the **Approving** activity. Also, indicate when a To Do is for an **Approval** activity.

2) Consolidate multiple application **Approval** List tables into a single platform table. Add support for structures based functions that can be used within workflow to determine workflow assignees. Structures which... ..properties of the ancestor of a point at a given layer.

Determine the assignee by getting the workflow dynamic properties of a point.

3) Incorporate **Approval** List Substitutes into ...is defined, To Dos that are already in the users To Do list are not affected

4) Provide a model window for consistent handling of **Approval** Status/Tracking. This would make sure that all the **Approval** Tracking windows have the same look and feel. It would also decrease the development cost of adding **Approvals** to new windows. The existing application tracking windows can still be used. The GUI is generally consistent.

5) Provide a PowerBuilder object/ancestor to support **Approve**, Reject and Comments from within an application window which will allow a view of the object being approved. This change would allow the **approver** to see the object that they are **approving**. Currently, **approvers** use the **Approval** Tracking window and have to Zoom to the actual object. For this release, the view of the actual object will be read-only.

6) Allow users to **Approve** items directly from the Task Details list without actually entering the application activity.

7). Provide workflow functions which will consolidate code so that an **Approval** Enabled window can use two functions to do ALL of the work required including

Generating rows for the tracking/status instance table for each **approver** on the user selected **approval** list

Generating To Dos for the first level of **approvers**

Generating any additional To Dos for Next Steps attached to the **Approval** enabled event

Returning the ID and the type of the **approver** used or an indicator that no **approval** process was required

New workflow functions may be written to support **Approvals**. Both are a variation of `pam0011(underscore)trig(underscore)event()`. The first, `pam0012(underscore)trig(underscore)event()`, will return a code indicating **Approval** Complete and will require three (3) additional arguments, `tracking(underscore)SQL(underscore)string`, `approver(underscore)used` and `approver (underscore)type(underscore)used`.

The flow of the new function will be:

- \* Do the same things that the old `trig(underscore)event` function does PLUS
- \* If the **approval** flag is set in a `next(underscore)step` then evaluate the `next(underscore)step(underscore)options` conditions to get the **Approver** ID and **approver(underscore)type(underscore)code**.
- \* If no **approval** is required, i.e. no list value is present, return 1, an indicator that the **Approval** is complete.
- \* Otherwise...

Update the **approver(underscore)used** and **approver(underscore)type(underscore)code** arguments for return to the caller .

- \* Using the application window provided string containing `svr(underscore)db(underscore)owner.stored(underscore)proc` and the formatted key values, insert rows into the Tracking table. Resolve Structures functions to provide the User/Workgroup/Role ID and the **Approval** Level for each **approver**. Set the status to N.

The second, `pam0013(underscore)notify(underscore)approvers()`, will generate the next batch of **Approval** To Dos as specified by the `apprvl(underscore)level` and will require one (1) additional argument, `next(underscore)approver (underscore)SQL(underscore)string`. It can also be used in the **Approval** window.

With the prior system, when an **approval** process is associated with an activity:

- \* From `dbupdate` event, call `update(underscore)insert(underscore)tran()` to insert a row into the database for the object to be approved.
- \* error handling

call `pam0011(underscore)trig(underscore)am(underscore)event(insertevent,... )`

select the owners from the **approval** list

build a `SQL(underscore)string` for each **approver** for the insert stored procedure for the instance table

insert a row into the instance table for each **approver**

call `pam0011(underscore)trig(underscore)am(underscore)event(sendapprmsg,... )` for each **approver** in the first batch.

\* Using the new functions, call update(underscore)insert(underscore)tran() to insert a row into the database for the object to be approved

error handling

```
complete(underscore)var = pam0012(underscore)trig(underscore) approval(underscore)event
(insertevent ,,,,SQL(underscore)string, approver(underscore)used,
approver(underscore)type(underscore)code) where SQL(underscore)string =
am(underscore)server(underscore)db(underscore)owner(underscore)g +
insert(underscore)instance(underscore)stored(underscore)procedure + &
instance(underscore)keys (formatted for SQL)
```

```
pam0013(underscore)notify(underscore)approvers(sendmsgappr, .. .., SQL(underscore)string,
approval(underscore)level) where SQL(underscore)string =
am(underscore)server(underscore)db(underscore)owner(underscore)g +
date(underscore)instance(underscore)stored(underscore)procedure + & instance-keys
(formatted... ..the values for the tracking/status table are the last parameters passed in.
```

The description below provides additional guidance and notes in the implementation of **Approvals**.

### Approval Table Design

The following tables change in the present invention to support **Approvals**:

next(underscore)step

\* The apprvl(underscore)ind column indicates whether the next(underscore)step is for an **Approval**. The mapping tool and the workflow engine handle **Approvals** slightly differently from other To Dos (see below).

message(underscore)queue(underscore)1

\* The apprvl(underscore)ind column is added to indicate that the To Do is an **Approval To Do**. The Task Detail window will allow **Approval** from there if TRUE.

The following tables are added :

apprvl(underscore)enabled(underscore)activity

The apprvl(underscore)enabled(underscore)activity table contains the IDs of Activities that are **Approval Enabled**. It contains the following columns:

\* activity(underscore)id (primary key) - The ID of an activity that has been coded to support customized **Approvals**.

\* `apprvl(underscore)event(underscore)id` (primary key) - The ID of an event which can generate an **Approval To Do**.

\* `reapprvl(underscore)event(underscore)id` - The ID of an event which will require reapproval because of changes to the object being approved.

\* `apprvl(underscore)activity(underscore)id` - The ID of the activity where the **Approval** should be performed, i.e. the Next Step activity.

\* `task(underscore)detail(underscore)apprvl(underscore)id` - The name of the PowerBuilder object that contains the application code to be used by Task Detail to support Approvals for this activity.

`apprvl(underscore)list`

The `apprvl(underscore)list` table contains **Approval** List IDs associated with the activities that use them. It contains the following columns:

\* `apprvl(underscore)list(underscore)id` (primary key) - The ID of an **Approval** List.

\* `apprvl(underscore)activity(underscore)id` (primary key) - The ID of an activity that has access to this list for **Approvals**.

\* `notes` - Text used as needed.

`apprvl(underscore)list(underscore)detail`

The `apprvl-list(underscore)detail` table contains a list of users, groups, roles or structures functions which can be associated with an **Approval** process. It contains the following columns:

\* `apprvl(underscore)list(underscore)id` (primary(underscore)key) - The ID of the **Approval** List whose members are defined here.

\* `apprvl(underscore)activity(underscore)id` (primary key) - The activity that has access to this list.

\* `apprvl(underscore)level` (primary key) - Indicates the order in which **Approvals** are performed.

\* `apprvr(underscore)id` (primary key) - The user, workgroup or role ID of this **approver** or the default **approver** should a structures function fail to be resolved.

\* `apprvr(underscore)type(underscore)code` (primary key) - Indicates whether **approver(underscore)id** is a user, workgroup, role or structure function.

\* `structure(underscore)function(underscore)type` - Indicates the type of structure being used.

\* `structure(underscore)group...` ...column ID of the data being used to resolve this structure function.

\* layer(underscore)name - The name of the Layer.

\* default(underscore)type - Indicates whether **approver** (underscore)id is a user, workgroup, role. Relevant when the structure function cannot be resolved and the **approver**(underscore)id is used as a default.

#### substitute

The Substitute tables contains a list of users who have substitutes defined for them. It contains the... ..number and datatype of the keys will vary from application to application.

\* instance key(s) (primary key)

\* apprvl(underscore)level (primary key)- Order in which **approvals** are performed.

\* apprvr(underscore)id (primary key)- The user, workgroup or role ID of this **approver**.

\* apprvr(underscore)type(underscore)code (primary key) - User, Workgroup or Role

\* apprvl(underscore)status(underscore)code - N = None (hasnt been sent a message to review...  
...Date of the last status change

\* actual(underscore)apprvr(underscore)id - ID of the user who performed the status update.

#### End User Windows

##### Activity Window **Approve/Reject** Standard

The new activity will inherit from the original activity to provide a complete, read-only view of whatever is being Approved. Functionality includes **Approve**, **Reject**, **Comments**, **Approval** Complete, generation of next level To Do messages and updates to the instance tracking table. The Yellow Sticky object is moveable so that all portions... ..seen.

#### Task Detail

plt0040(underscore)todo needs to be modified so that when the message queue row indicates that the Next Step activity is an **Approval**, an additional button will be displayed. When rows from the detail list are selected, enable the button. If **Approval** button clicked, execute the application provided code to update the **Approval** Tracking, generate trig(underscore)events and complete the **approval** process as needed. A reusable PowerBuilder object will contain the necessary application code.

#### User Substitute

The Substitute window allows a user to define a substitute... ..with the system so that an informational To Do is generated when a user is made a substitute for another user.



## Administration and Tracking Windows

### Approval List Definition

This window allows the administrator to create an **Approval** List. Each list is keyed by the list name and the activity ID for which it is applicable. A Popmenu and response window support Copy from one List to another.

### Tracking/Status Ancestor

This window is keyed by the application key(s). It shows the current status of the **Approval** process for the item shown. It also shows a list of the future **approvers** and allows the current user to **Approve** or **Reject** if appropriate. PopMenu and Tool Bar provide Accept, Reject and Zoom to Document actions.

### Administrator Substitute

The Administrator Substitute window is a simple... ..are assign(underscore)to type and user or workgroup.

Rule: Add, change, delete of Substitutes does NOT impact existing workflow.

### Workflow Mapping Tool

When an **Approval** Enabled activity is mapped, the map knows which events are **Approval** Enabled. When the Insert event is chosen from the windows list of available events, the **Approval** Activity is presented as a possible next step. When chosen the windows presentation is slightly different in that **Approval** Lists are presented for the assignment of To Do messages.

### Application Impact Issues

**Approval** Enabled activities must:

1) Have moved **Approval** Lists and Substitutes to the common Platform tables. Note: Substitutes will now apply to all workflow NOT just **Approvals**. These tables and their respective maintenance windows will replace the applications' individual implementations. This means that two windows can be dropped for each **Approval** Enabled activity.

2) Use the new TrigEvent function to manage **approval** processing and to process "Approval Complete" code, if the user has chosen to NOT **approve** under certain conditions. This function will return an indicator of **Approval** Complete and the ID of the **Approver** and its type code which was used to generate the Tracking Table rows. The application window will provide a string containing svr(underscore)db(underscore)... ..proc and the formatted key values to Insert rows into the Tracking table. The workflow engine will provide the User/Workgroup/Role ID and the **Approval** Level for each **approver**. Structures functions will be resolved at this time.

This means that some code can be removed from an activity window. Each application designer should consider whether it would be useful to store the name of the **Approver** used.

3) Have written the PowerBuilder function which will process **Approvals** from the Task Detail window. A model will be provided and an attempt will be made to require the bare minimum of code from the... ..the-blanks with code that currently exists in other places.

Other steps which could be taken to move in the direction of an internally consistent **Approval** process are:

- 1) Rewrite Tracking/Status windows to use new standard.
- 2) Use the model for **Approve** /Reject from the document in addition to the **Approve** from the Tracking window.

#### StreamBuilder Impact Issues

##### **Approval** Activity window

Users who build activities which they wish to **Approval** Enable will need a GUI to enter and maintain the `apprvl(underscore)enabled(underscore)activity` table. **Approval** Enable one of the existing Sample Application windows. (`psa0800(underscore)invoice`). Use the new Yellow Sticky to do the **Approval** on a read-only view of the original invoice using the new model. (`psa0810(underscore)invoice(underscore)approval`).

##### **Approval** Tracking Window

Provide a sample of the **Approval** Tracking/Status implementation using the new Ancestor. (`psa0850(underscore)invoice(underscore)status`).

##### Task Detail **Approval** object

Provide a sample of the PowerBuilder function used to **Approve** from the Task Detail window. (`psa0800(underscore)invoice(underscore)approve`).

##### Dependencies

1) Structures integration requires that all structures that are used within workflows be replicated to all work flow servers 110 in the system.

2...

DIALOG(R)File 348: EUROPEAN PATENTS  
(c) 2011 European Patent Office. All rights reserved.  
8/3K/3 (Item 3 from file: 348)  
00377891

### Electronic document approval system

Elektronisches System zum Genehmigen von Dokumenten  
Systeme electronique pour approuver des documents  
Electronic document **approval** system

#### Patent Assignee:

- **International Business Machines Corporation** (200120)  
Old Orchard Road; Armonk, N.Y. 10504 (US)  
(applicant designated states: DE;FR;GB;IT)

#### Inventor:

- **Lemle, Philippe**  
Le Bel Vue B 77, avenue du Groupe Morgan; F-06700 Saint Laurent du Var; (FR)

#### Legal Representative:

- **de Pena, Alain et al (15151)**  
Compagnie IBM France Departement de Propriete Intellectuelle; F-06610 La Gaude;  
(FR)

	Country	Number	Kind	Date	
Patent	EP	387462	A1	19900919	(Basic)
Patent	EP	387462	B1	19960508	
Application	EP	89480045		19890314	
Priorities	EP	89480045		19890314	

#### Designated States:

DE; FR; GB; IT

#### International Patent Class (V7): G06F-017/21; ;

**Abstract** ...terminal connected to the system network to select a form among prestored document forms, fill said form in and then have said form mailed for **approval** by system users selected based on predefined and stored rules. The **approval** path is being permanently updated by the

system. The system is made to filter access to the filled-in forms using prestored tables, and monitor the mailing and processing said filled-in forms for **approval**.

**Abstract Word Count:** 95

**Language Publication:** English

**Procedural:** English

**Application:** English

Fulltext Availability Available Text	Language	Update	Word Count
CLAIMS A	(English)		711
SPEC A	(English)		9253
CLAIMS B	(English)	EPAB96	838
CLAIMS B	(German)	EPAB96	909
CLAIMS B	(French)	EPAB96	1048
SPEC B	(English)	EPAB96	9365
Total Word Count (Document A) 9965			
Total Word Count (Document B) 12160			
Total Word Count (All Documents) 22125			

**Specification:** ...A1

## ELECTRONIC DOCUMENT **APPROVAL** SYSTEM

### Field of the Invention

This invention deals with **approval** on contents of electronically generated and mailed documents and with a system for generating, monitoring and processing said documents.

### Background of Invention

Electronic mailing systems... ..in European application No. 0,269,875 wherein shell forms are processed.

Once the form is completed, the originator has to get it approved. The **approval** process can be simple (for instance **approval** by the originator's manager), in that case, a simple electronic mail forwarding operation would do the job.

But in most instances the requirements are more complex and several levels of management or functional **approvals** may be required, e.g. by a Financial Analyst, a Budget Controller, etc... Often the **approval** process depends on data filled into the form: e.g. for a **purchase order** if the amount of a purchase requested doesn't exceed a given value, one level of management is sufficient, otherwise two levels of management are necessary, for instance. Also because of

management decisions, **approval** rules for a given form can be changed without the form itself being changed.

With a conventional system, when the originator passes the document to the first **approver** (e.g. his manager) to get his signature (i.e. **approval**), he has lost control of it. He cannot be sure where the document is or who has it or if the document is hand carried or forwarded through internal mail. This prevails for any step in the **approval** process : first **approver** loses control upon transferring the document to the second **approver** and so on. Often **approvers** have to keep a paper copy of the document before passing it to the next **approver**.

In some instances **approval** has to be given by a delegate, but people may not know who the delegate is.

The **approver** list may also need to be modified during the **approval** process : an **approver** can be replaced by his manager, the order of **approval** can be changed or more information may be requested by an **approver** from another **approver** who has already signed.

Finally, all documents issued from forms and approved reach the person or the department who has to process and execute the request. First, checking must be done : data checking and **approval** process checking.

Then, if all is correct, action is taken such as keying the data into an operational system.

Generally, no information is returned to... ..satisfied, or he has to get information by phone or mail.

The document are to be kept not only during all the time the involved **approval** process is running, but also after that, during a retention period fixed for each form.

All known systems, however, lack efficient means for monitoring filled-in forms (herein referred to as documents), **dynamically** and electronically computing any **approval** path to be followed by a specific form based on its contents when filled-in, and controlling said path as well as filtering any request... ..object of this invention is to provide a system for accessing a prestored blank form library, selecting a form, filling-in said form, computing an **approval** path based on filled-in form data and on specific predefined **approval** rules referring to user's job or function within the population of system's attached users, and monitoring and controlling the corresponding **approval** operations.

Another object is to provide a system for filtering any request for access to any filled-in form (document) based on filled-in data, stored updatable **approval** rules and requestor's identity.

In other words, this invention addresses the automation of all the steps involved in the processing of documents whose contents require complex **approvals**.

That includes documents origination, **approver** list determination, electronic signatures (i.e. **approval**) authentication, finalization operations, storage and general follow-up of the process.

More particularly the invention addresses an **approval** system for controlling the processing of a user originated document requiring signature by electronic **approval** by system selected users, in an electronic mailing system including terminals attached to a digital network, virtual machines (VM) including computer means, memory and software... ..the population of system attached users, and means for generating processing and monitoring electronic documents to be mailed from any terminal to any user, said **approval** system including:

- means for storing and updating function tables wherein each system user's function and address are identified;
- means for storing document forms;
- means for storing predefined **approval** rules based on user function document type and document forms contents;
- terminal controllable means for selecting, accessing, filling- in, processing and mailing any selected form whose contents is to be subjected to **approval**;
- means sensitive to said mailing for addressing said function tables and, based upon said **approval** rules, for determining the **approval** path among the system attached users; and,
- means sensitive to said **approval** path determination for monitoring the mailing and processing of said filled-in form accordingly.

These and other objects, characteristics and advantages of the invention will... ..the programs; and

- a System-disk which contains all necessary routines for a user to run the SEALING application, i.e. routines needed by the **approval** application and further defined in this description. When a user wants to run the application, he has to make a read-only link to this...will be made apparent in the following description. But, it may already be stated that it contributes to provide a higher security level to the **approval** process. To that end, the only SQL commands the end user may use are predefined into access modules stored into the SEALDBA machine attached disks... ..have been designed and stored in the system (SEALSYST) for further use and conversion into documents to be processed (e.g. approved) using the invention.

**Approvers** are normally designated by reference to their function, e.g. manager first, second, ... line of department No. xx. The function may be delegated, in which... ..different from the original assignee (titular) of the function.

When the system user (originator of a considered document) wishes to forward the document for proper **approval**, the **approval** path is **dynamically** computed using the contents of specific data fields within the document, and predefined **approval** rules. In the following implementation, the **approvers** may have been split into two categories : "Authorizers" and "Reviewers". Only an authorizer may accept (concur) or reject a document. A reviewer can only give an advice. A

negative reviewer's opinion requires a subsequent authorizer **approval** for the document to proceed.

Finally, after being processed by the last **approver**, the document is forwarded automatically to a finalizing VM machine performing conventional operations such update, format and if required encrypt and send through the network to another network node, and, for the purpose of this invention, perform a control operation tailored to ascertain a higher security level to the **approval** system.

The above operations are summarized in Figure 4, using the facilities made available through SEALDBA, SEALSYST as well as the VM user's machine.

It should be noted therein, that the document as well as any information (e.g. functions) required for **approval** are not forwarded from one **approver** to the next. They are accessed from the general data base dynamically. This architecture enables updating constantly the **approval** path to the company's personnel (users) moves or reassignments.

From a functional standpoint the software architecture is organized as represented in Figure 5. In other words, access to the **approval** system is provided by software tools in REXX language. These tools (programs) will access the library of SEALING General Purpose Programs as well as access... ..Data System General Information GH24-5064

- SQL/Data System Terminal User's Guide SH24-5045

Represented in Figure 6 is a symbolic representation of the **approval** system centered on an SQL/DS data base accessible on Read/Write basis throughout the process. An entry for a document preparation (as will be explained later) involves read/write operations into the data base. This process step is followed by an **approval** step which can be triggered from both the document preparing user or directly from an **approver**. Once approved the document is further processed, followed-up and executed as needed. For that purpose not only the SQL/DS needs be accessed, but... ..instance VM directories other wise available. The document is finally forwarded to storage. Also available are means for updating "function" data particularly useful to the **approval** process; and consulting/analysis means to be used for instance for performing statistical operations.

Represented in Figure 7, is a mapping of the system data... ..data can be split into general tables required to run the system and specific tables which are accessed and/or generated when needed by the **approval** process. Obviously, the contents of said tables are based on preselected **approval** criteria which could be changed and therefore are not limitative. The following description is made with reference to criteria implemented in the preferred embodiment.

(A ... ..can find:

1. The LOGON tables including users identification data, (e.g. nodeid; userid).

2. Tables related to **FUNCTIONS** : it is herein assumed that the **approval** system users are assigned specific functions governing the **approval** rules (e.g. managerial organization). Said tables include :

- **FUNCTION** tables defining existing functions and providing for each function an acting person and a titular person identification.
- **PREVIDEL** tables registering provisions of delegations, assuming **approval** competence could be delegated from one person to another.
- **HISTFUNC** keeps track of any modification occurring in function tables for audit purposes (new function, delegation, change of titular).

3. Tables related to **APPROVAL** process :

**APPFUTU**, **APPWAIT** and **APPDONE** relate to documents in progress.

- **APPFUTU** contains for each document the functions to be involved (in the future) in the **approval** process.
- **APPWAIT** contains for each document the functions awaiting for an action on the document.
- **APPDONE** contains for each document the functions having already acted...It has exactly the same structure as **APPDONE**.

4. Tables related to **DOCUMENTS** :

Filled-in forms provide so called documents to be subjected to the **approval** process. Each document includes at least a **HEADER** with the data defining the document. All headers are dynamically stored into **HEADERS** tables.

**COMMENTS** Tables contain for each document the comments added by the **approvers** during the **approval** process as they act on the document.

(B) In the specific Tables, one can find :

1. Tables dealing with **FUNCTIONS** :

- determination of function table coded... ..contain the specific document data.

Control tables are sometimes necessary to control the data entered to prepare a document or to modify these data during **approval** process. Once defined and updated, they can be used for several different types of documents.

Follow-up tables are sometimes necessary to give follow-up... ..hereunder : (Table omitted)



An important concept of the system is the concept of "function". In a company or any group of people within which the **approval** system is needed to operate, prerogatives, e.g. **approval** competence, are assigned based on job assignments or title, herein referred to as "Function".

In an **approval** process, it is not the signature of a specific person which is required, but the signature of the person presently assigned the required function.

A... ..as a sum of prerogatives acknowledged by the company and assigned to a person. Conversely, a given person may have several different functions.

The **approval** system of this invention considers functions assigned rather than people. There are a lot of advantages to use this criterium because functions are generally more stable than people. For instance, **approval** rules defined in a company are generally related to people's level within the company (hierarchy).

In the system, a "function" is completely defined by... ..referenced by managed department number)

The system provides an interactive means to manage any other function, as soon as this function becomes necessary in an **approval** process.

Examples of **APPROVAL TABLES** of Figure 4 are represented hereunder. (see image in original document) (see image in original document) APPHIST : same structure as APPDONE.

APPFUTU : contains for each document in the **approval** process the list of functions which will have to deal with the document later on.

In this table the document is uniquely identified by the... ..Typfun) and a reference number (e.g. Service or Department number (Reffun)). An order number 1, 2, 3 ... indicates which function(s) will need to **approve** next, assuming no changes to the **approver** list has occurred. Several functions can have the same order number.

**Approver** type (Apptyp) indicates if the function is in the list as an Authorizer (A) or as a Reviewer (R). The Authorizer actions could be "Authorize" or "Reject"; while a reviewer should "Approve" or "Disapprove".

An index indicates if the function is mandatory or not on the **approver** list. This will be important to make changes in the **approver** list. If the function is not mandatory, any **approver** can suppress it, but if it is mandatory, either it is not possible to suppress it, or another function must replace it.

APPWAIT contains for each document in **approval** process the function(s) which are actually waiting for the document (i.e. next to act).

As APPFUTU, it contains document identification, function identification, **approver** type and mandatory indicator. It contains also the previous **approver**'s name, one line personal comments from the previous **approver**, the date and time of previous action.

APPDONE : contains for each document the functions that have already acted on the document, the decisions and identification of the persons who have acted and of the titular if he is different.

As APPFUTU and APPWAIT, it contains document identification, function identification, **approver** type and mandatory indicator. It contains also the decision of the **approver** Y for Authorize or **Approve**; N for Reject or Disapprove,

the identification of the **approver**, name and employee serial number, the identification of the titular of the function, name and employee number, the delegation indicator, the action date and time.

As already mentioned, the **approval** system for the best mode of this invention is made to use predesigned forms. Any user wishing to start a request for **approval** operation, will access a document form (blank) and fill-it in. A form includes predefined fields which, when filled-in will be stored into a... ..identification, i.e. type and reference of document. Each document is assigned a status (one character) which can be :

P : document is in progress in **approval** process

F : document is finalized, **approval** process is over.

S : document has been sent to an operational system

T : document has been transmitted for action (acknowledgment received from operational system).

H... ..his function (e.g. INDI for employee), his employee serial number and his name.

The table also contains the subject of the request submitted to **approval** (document) and the originating date and time.

In addition to the function tables already mentioned the system will use Specific Function Tables storing data to be used for defining the **approval** rules selected by the form user's company.

SPECIFIC FUNCTION TABLES (see image in original document) (see image in original document)

These tables allow retrieving a function reference from a parameter. They will be used during **approver** list determination.

The example shows so called Determination Tables useful to determine the manager who is responsible for a documented project.

Function CHARACTERISTIC tables enable determining parameters from the reference of a given type of function. They will be used during **approver** signature validation. In other words, these tables store predefined **approval** rules.

The example shows the amount of expenses authorized based on the considered manager's function and the company's selected rules setting expense thresholds.

From the above considerations, one may understand that any filled-in form (=document) contents is dynamically used by the system to determine the **approval** path when considered in combination with **approval** rules. In addition, the system is designed to enable adding or modifying forms, as needed, in a fairly simple way. Therefore, the various types of... ..the flow chart of Figure 9).

First, the user is recognized for being logged on a virtual machine which the system knows as a "signature" ( **approval**) machine assigned to a registered user.

The FUNCTION table is the main filter to access the documents. Said table is permanently updated to reflect the... ..used to find documents on which action has been taken already. Documents partially approved are stored in table APPDONE, while those approved by all required **approvers** (or rejected) are stored in an APPHIST or historic table.

To make the system attractive and end-user friendly from an operational standpoint, panels have...menu 1 calls PREPARE EXEC. (See Fig. 10 for a high level flow chart of the process for preparing a document to be submitted to **approval**).

This EXEC starts displaying the list of forms available to the user in a "Choose a Form Category" menu. It should be noted that the system is made to process forms of various categories like "purchasing" orders (APPR), "financial" orders (FINA), requests for getting **approval** to tailor a VM Machine to a new user (LOGO), etc... (see image in original document)

If there are too many categories to fit on... ..system checks in this case that the reference exists in the data-base and that the user is either the originator, or one of the **approvers**. Thus filtering access to said existing document.

The layout of the screens which the user is presented with, depends on the original form designer's...to the header screen

- PF3 Repeat the displayed item to create a new one
- PF4 Add a blank item
- PF5 All operations before sending in **approval** (see below)
- PF10 Display next item if it exists
- PF11 Display previous item if it exists

If the user presses PF12, the system shows the... ..image in original document)

From here, the user will be able to :

- PF1 : View the document as it can be viewed later by all the **approvers**
- PF2 : Change the document data
- PF4 : Add free comments which will be viewed to all the **approvers**
- PF5 : Prepare sending in **approval** (see below)
- PF6 : Save the data entered as a draft document (and retrieve it later)
- PF8 : Print the document. A print image of the document... ..self explanatory flow-chart of Figure 10.

Down to this point the process lead to a fully prepared document ready for being submitted to the **approval** process. Therefore, if the user presses PF5 either from the data entry panels, or from the "Process prepared document", the system performs the following operations... ..found, the system displays again the data entry panel where the field bearing an erroneous data is indicated by the cursor. Then, it determines the **approval** path based on functions involved, specific rules assigned for the type of document involved, and document data (see Figure 14).

If an error is found, the message "Unable to determine **approval** process" is displayed and no action is performed.

Finally, it determines for each function of the **approval** process, the acting person at the present time and the titular.

The result is displayed (see Figure 15) to the considered user (originator) as shown... ..document)

If the user presses PF12, he just returns to "Process the document" Menu.

Otherwise, by depressing PF1 he will be able to change the **approver** list with some controls and restrictions (see below with reference to Figures 15 and 16).

If he does press PF7, the document is created and waits for action of the first function(s) in the **approval** process (see Figures 15 and 16).

Represented in Figure 11 is a flow-chart summarizing the operations achievable on an already filled-in document. The... ..functional key labeled "Open the Mail", assuming SEALING documents are waiting action. The message looks like:

"You have 5 documents awaiting your action in Electronic **Approval** System. Do you wish to process these ? Type Y and press ENTER if you wish". The system then shows first a list of categories of... ..document)

The user can see on this screen the document type and reference, the subject, the originator's name, who was the 1 last previous **approver** and personal comments from this previous **approver**. Also mentioned is the function the present user is asked to act for, and eventually the owner for the function if the user is a delegate.

By depressing PF1, the user can view all information about the document, i.e. the document itself; the **approver** list with decisions of **approvers** who have already acted on the document; the document originator and **approvers** comments, if any.

Depressing PF4, enables the user adding his own comments to the other **approver's** comments after acting on the document.

PF5 is not available for all **approvers**. This PF Key is only active if the document is originally tailored to authorize the function to add or modify data. In this case, the user is shown data modification panels and can add or modify data in the document. These modifications can affect the **approval** path process.

PF7 must be used to act on the document.

If the user is an Authorizer, he will see the screen below. (see image in original document)

The user can change the **approvers** list by pressing PF7 (seen later).

The user can Authorize (**Approve**) the document with PF1. The system will display a confirmation panel which looks as follows. (see image in original document)

The next screen shows the next **approver** and allows the user to type one line of personal comments for said next **approver's** attention.

If the user is the last **approver**, the confirmation panel is different and looks as follows : (see image in original document)

Authorization means finalization of the document.

An Authorizer can also Reject... ..his decision, indicating that in case of non confirmation, the document will be rejected, an information will be sent to the originator and to each **approver** who has already acted on the document.

The user may also choose PF3 to request additional information from another **approver**.

In this case, the user has to choose an **approver** in the screen below. (see image in original document)

He can choose the originator, an **approver** who has already acted on the document or an **approver** who has not yet seen the document, then he gets a confirmation panel as below. (see image in original document)

If the user confirms, the document is available for the chosen **approver** and the user will get it back again after this **approver** has acted upon said document. If the user is a Reviewer in the **approval** process, he is shown the following screen. (see image in original document)

There will be two different confirmation panels no matter whether the user is the last **approver** or not. Request for additional information operation is quite the same as requested to an Authorizer. It should be remembered that a Reviewer cannot reject a document. He can just disapprove the document, in which case a further Authorizer **approval** is required. The system determines if there is an authorizer in the list of next **approvers**. If there is no Authorizer in the list of next **approvers**, the system asks the user to choose among the Authorizers who have already acted on the document and will send the document back to the... ..Or the document has been rejected by an Authorizer or has been cancelled by the originator and the user is either the Originator or an **Approver** who has already acted on it.

PF4 has just the effect to cancel reference reminder to said documents. The user can always access the document... ..can access documents using several alternatives (see Figure 12):

- PF1 : Lets the user access the documents he has originated and which are currently in the **approval** process.

NB: by user one means the person assigned corresponding function.

- PF2 : Lets the user access the documents he has processed and which are currently in the **approval** process.

- PF3 : Lets the user access all the documents, whatever their status (in progress, finalized, rejected, cancelled...) he has originated between two dates the user...the status is "In progress", the user is presented with the screen below. (see image in original document)

The user can

- view the document (data, **approvers** list and comments).
- copy the document to create a draft
- print the document

If he is the originator, he can also cancel the document. In... ..user is presented the following confirmation panel. (see image in original document)

Pressing PF7 in "Process the document found" menu allows the originator or an **approver** who has already acted on the document to change the **approver** list as he could have done when originating the document or acting upon the document.

When changes are prepared, the system displays the following panels. (see image in original document)

If all changes are correct, the user can press PF1 and the document proceeds with the new **approver** list.

PF12 will cancel all the changes.

If the status of the document is not "In progress", the available actions are different as shown hereafter. (see image in original document)

The user can

- view the document (data, **approver** list and comments).
- copy the document to create a draft
- print the document

Follow-up information may also be given when the user presses PF4... ...looks for documents awaiting action and give the references. For obvious security purposes, no further information about document can be obtained.

As already mentioned an **approver** designation, could be delegated from one user to another under predefined conditions. The user can access this part of the system by pressing PF7... ...In this case, the system considers the user as absent. This feature will be used to bar access to the document for instance when an **approver** is acting on it. So, originators and **approvers** will be informed and will be able to react to this situation.

To validate his entries, the user has to press the enter key.

Using... ...the transfer.

Having thus described from a technical as well as functional standpoint, the means involved in this invention, one may therefore fully comprehend the **approval** system operation. **Approval** system operations has however been summarized in Figures 14 through 17, and Function management in Figure 18.

Represented in Figures 14 through 16 are, the means involved in the determination of the list of **approvers**. Obviously, the system has been limited to simple cases to simplify explaining the operation. One assumes the filled-in document (e.g. **Purchase Order**) includes a Project code, a Purchaser code and an Amount of expenses set by the purchase requesting user. The software

means uses the ... FaMANAD2) needs be addressed. Also, due to the type of document involving expenses, the logic adds an Investment Responsible (INVT) to the list of required **approvers**. Same applies to Purchaser (PURC). The system loads the information into two separate tables located in the document originator VM user's machine memory designated as FUTU and DONE respectively. The DONE table contains so called "shadow" **approvers** or virtual **approvers** whom the system will enable read-only access to the corresponding document. As already mentioned, once initiating the **approval** process, the originator may amend the list of **approvers** up to a certain extent based on predefined rules. For instance, deletion of a first line manager from the list of **approvers** will trigger automatic insertion of the second line manager, and so on.

Then, the originator sending decision has the effect to unload the first **approver** references from FUTU table into a NEXTWAIT table while the others are loaded into a NEXTFUTU table in the preset ordered list, both tables NEXT... ..and updating the corresponding SQL data base tables, i.e. APPFUTU, APPWAIT and APPDONE of the SEALBDA machine.

The SEALING system also controls mailing **approval** requests to designated **approvers** whose action are controlled as represented in Figure 16.

The designated **approver** in NEXTWAIT gets access to the data in the corresponding SQL/DS tables in its own VM machines into FUTU, WAIT and DONE tables. Then NEXTFUTU, NEXTWAIT and NEXT DONE are set from FUTU, WAIT and DONE tables respectively. These tables contents are used by current **approver** to perform and record the following operations:

- **Approver** list modification: the **approver** may amend the list as the originator was able to do.
- Modification to **approval** process: the system controlled by any amendment to the data of document due to current **approver**, amends the **approval** path accordingly, and
- **Approver** validation: compliance test with the **approver's** designation rules is performed.

Tables updating are performed, i.e. once current **approval** is executed, a shift is operated with next **approver** in NEXTFUTU being shifted into NEXTWAIT.

Once forwarded, the current **approval** data are used to update the SQL/DS tables through add, modify or delete operations. They are then available for next **approver's** action and so on.

A fairly high security level has been provided in this invention to limit false **approvals** due either to human error or to voluntary operation. First, one should notice that **approval**, i.e. insertion of a "Y" or "N" flag into a predetermined field characterizing the considered document is inserted into the SQL/DS Table APPDONE otherwise accessible to the user on a Read-Only basis (see MYSIGNAT in Figure 6). Second, said "signature" or **approval** insertion is operated only upon execution of a predetermined SQL command involving a signature validity check. The flow chart of such a signature validation operation... ..The SQL command triggered by the MYSIGNAT order accesses various SQL tables to gather data stored therein to ensure that the user presently trying to **approve** or **disapprove** is entitled to do so. First APPWAIT table is



accessed to ensure that the considered document defined by typdoc/refdoc is waiting therein...  
...Table, together with data and time are enabled upon a positive signature validation test result.

As mentioned, the population of users attached to the same **approval** system is managed in an unique SQL table, i.e. the FUNCTION Table, as represented in Figure 18. For each function, i.e. Manager, Purchaser...

### **Specification: ...B1**

#### Field of the Invention

This invention deals with **approval** on contents of electronically generated and mailed documents and with a system for generating, monitoring and processing said documents.

#### Background of Invention

Electronic mailing systems... ...in European application No. 0,269,875 wherein shell forms are processed.

Once the form is completed, the originator has to get it approved. The **approval** process can be simple (for instance **approval** by the originator's manager), in that case, a simple electronic mail forwarding operation would do the job.

But in most instances the requirements are more complex and several levels of management or functional **approvals** may be required, e.g. by a Financial Analyst, a Budget Controller, etc... Often the **approval** process depends on data filled into the form: e.g. for a **purchase order** if the amount of a purchase requested doesn't exceed a given value, one level of management is sufficient, otherwise two levels of management are necessary, for instance. Also because of management decisions, **approval** rules for a given form can be changed without the form itself being changed.

With a conventional system, when the originator passes the document to the first **approver** (e.g. his manager) to get his signature (i.e. **approval**), he has lost control of it. He cannot be sure where the document is or who has it or if the document is hand carried or forwarded through internal mail. This prevails for any step in the **approval** process : first **approver** loses control upon transferring the document to the second **approver** and so on. Often **approvers** have to keep a paper copy of the document before passing it to the next **approver**.

In some instances **approval** has to be given by a delegate, but people may not know who the delegate is.

The **approver** list may also need to be modified during the **approval** process : an **approver** can be replaced by his manager, the order of **approval** can be changed or more information may be requested by an **approver** from another **approver** who has already signed.

Finally, all documents issued from forms and approved reach the person or the department who has to process and execute the request. First, checking must be done : data checking and **approval** process checking.

Then, if all is correct, action is taken such as keying the data into an operational system.

Generally, no information is returned to... ..satisfied, or he has to get information by phone or mail.

The document are to be kept not only during all the time the involved **approval** process is running, but also after that, during a retention period fixed for each form.

Disclosed in the Patent US-A-4,503,499 is... ..disclosed in Patent Abstracts of Japan, Vol. 9, No 206(P/149), of October 19, 1982, is a system wherein the document meant to receive **approval** by circulation is inputted together with circulation order facilities.

All known systems, however, lack efficient means for monitoring filled-in forms (herein referred to as documents), then computing fully dynamically and electronically the specific and correct **approval** path to be followed by each considered form based on its contents when filled-in, with reference to predefined sets of **approval** rules to be combined with varying titles or hierarchical positions of respective users to be automatically selected within the system users population, and controlling said... ..object of this invention is to provide a system for accessing a prestored blank form library, selecting a form, filling-in said form, computing an **approval** path based on filled-in form data and on specific predefined **approval** rules referring to user's job or function within the population of system's attached users, and monitoring and controlling the corresponding **approval** operations.

Another object of the invention is to provide a system for filtering any request for access to any filled-in form (document) based on filled-in data, stored updatable **approval** rules and requestor's identity.

In other words, this invention addresses the automation of all the steps involved in the processing of documents whose contents require complex **approvals**.

That includes documents origination, **approver** list determination, electronic signatures (i.e. **approval**) authentication, finalization operations, storage and general follow-up of the process.

More particularly the invention, as claimed, addresses an **approval** system for controlling the processing of a user originated document requiring electronic **approval** by system selected ...the population of system attached users, and means for generating processing and monitoring electronic documents to be mailed from any terminal to any user, said **approval** system including:

- means for storing and updating function tables wherein each system user's function and address are identified;

- means for storing document forms;
- means for storing predefined **approval** rules based on user function document type and document forms contents;
- terminal controllable means for selecting, accessing, filling-in, processing and mailing any selected form whose contents is to be subjected to **approval**;
- means sensitive to said mailing for addressing said function tables and, based upon said **approval** rules, for determining the **approval** path among the system attached users; and,
- means sensitive to said **approval** path determination for monitoring the mailing and processing of said filled-in form accordingly.

These and other objects, characteristics and advantages of the invention will...the programs; and

- a System-disk which contains all necessary routines for a user to run the SEALING application, i.e. routines needed by the **approval** application and further defined in this description. When a user wants to run the application, he has to make a read-only link to this... ..will be made apparent in the following description. But, it may already be stated that it contributes to provide a higher security level to the **approval** process. To that end, the only SQL commands the end user may use are predefined into access modules stored into the SEALDBA machine attached disks... ..have been designed and stored in the system (SEALSYST) for further use and conversion into documents to be processed (e.g. approved) using the invention.

**Approvers** are normally designated by reference to their function, e.g. manager first, second, ... line of department No. xx. The function may be delegated, in which... ..different from the original assignee (titular) of the function.

When the system user (originator of a considered document) wishes to forward the document for proper **approval**, the **approval** path is **dynamically** computed using the contents of specific data fields within the document, and predefined **approval** rules. In the following implementation, the **approvers** may have been split into two categories : "Authorizers" and "Reviewers". Only an authorizer may accept (concur) or reject a document. A reviewer can only give an advice. A negative reviewer's opinion requires a subsequent authorizer **approval** for the document to proceed.

Finally, after being processed by the last **approver**, the document is forwarded automatically to a finalizing VM machine performing conventional operations such update, format and if required encrypt and send through the network to another network node, and, for the purpose of this invention, perform a control operation tailored to ascertain a higher security level to the **approval** system.

The above operations are summarized in Figure 4, using the facilities made available through SEALDBA, SEALSYST as well as the VM user's machine.

It should be noted therein, that the document as well as any information (e.g. functions) required for **approval** are not forwarded from one **approver** to the next. They are accessed from the general data base dynamically. This architecture enables updating constantly the **approval** path to the company's personnel (users) moves or reassignments.

From a functional standpoint the software architecture is organized as represented in Figure 5. In other words, access to the **approval** system is provided by software tools in "Restructured Extended Executor (RSXX)" language. These tools (programs) will access the library of SEALING General Purpose Programs as... ..Data System General Information GH24-5064

- SQL/Data System Terminal User's Guide SH24-5045

Represented in Figure 6 is a symbolic representation of the **approval** system centered on an SQL/DS data base accessible on Read/Write basis throughout the process. An entry for a document preparation (as will be explained later) involves read/write operations into the data base. This process step is followed by an **approval** step which can be triggered from both the document preparing user or directly from an **approver**. Once approved the document is further processed, followed-up and executed as needed. For that purpose not only the SQL/DS needs be accessed, but... ..for instance VM directories otherwise available. The document is finally forwarded to storage. Also available are means for updating "function" data particularly useful to the **approval** process; and consulting/analysis means to be used for instance for performing statistical operations.

Represented in Figure 7, is a mapping of the system data... ..data can be split into general tables required to run the system and specific tables which are accessed and/or generated when needed by the **approval** process. Obviously, the contents of said tables are based on preselected **approval** criteria which could be changed and therefore are not limitative. The following description is made with reference to criteria implemented in the preferred embodiment.

(A... ..can find:

1. The LOGON tables including users identification data, (e.g. nodeid; userid).
2. Tables related to FUNCTIONS : it is herein assumed that the **approval** system users are assigned specific functions governing the **approval** rules (e.g. managerial organization). Said tables include :

FUNCTION tables defining existing functions and providing for each function an acting person and a titular person identification.

PREVIDEL tables registering previsions of delegations, assuming **approval** competence could be delegated from one person to another.

HISTFUNC keeps track of any modification occurring in function tables for audit purposes (new function, delegation, change of titular).

### 3. Tables related to **APPROVAL** process :

APPFUTU, APPWAIT and APPDONE relate to documents in progress.

APPFUTU contains for each document the functions to be involved (in the future) in the **approval** process.

APPWAIT contains for each document the functions awaiting for an action on the document.

APPDONE contains for each document the functions having already acted... ..It has exactly the same structure as APPDONE.

### 4. Tables related to DOCUMENTS :

Filled-in forms provide so called documents to be subjected to the **approval** process. Each document includes at least a **HEADER** with the data defining the document. All headers are dynamically stored into **HEADERS** tables.

**COMMENTS** Tables contain for each document the comments added by the **approvers** during the **approval** process as they act on the document.

(B) In the specific Tables, one can find :

#### 1. Tables dealing with **FUNCTIONS** :

determination of function table coded... ..contain the specific document data.

Control tables are sometimes necessary to control the data entered to prepare a document or to modify these data during **approval** process. Once defined and updated, they can be used for several different types of documents.

Follow-up tables are sometimes necessary to give follow-up... ..in original document)

An important concept of the system is the concept of "function". In a company or any group of people within which the **approval** system is needed to operate, prerogatives, e.g. **approval** competence, are assigned based on job assignments or title, herein referred to as "Function".

In an **approval** process, it is not the signature of a specific person which is required, but the signature of the person presently assigned the required function.

A... ..defined as a sum of prerogatives acknowledged by the company and assigned to a person. Conversely, a given person may have several different functions.

The **approval** system of this invention considers functions assigned rather than people. There are a lot of advantages to use this criterium because functions are generally more stable than people.

For instance, **approval** rules defined in a company are generally related to people's level within the company (hierarchy).

In the system, a "function" is completely defined by... ..referenced by managed department number)

The system provides an interactive means to manage any other function, as soon as this function becomes necessary in an **approval** process.

Examples of **APPROVAL TABLES** of Figure 4 are represented hereunder.

## **APPROVAL TABLES**

(see image in original document) (see image in original document)

APPHIST : same structure as APPDONE.

APPFUTU : contains for each document in the **approval** process the list of functions which will have to deal with the document later on.

In this table the document is uniquely identified by the... Typfun) and a reference number (e.g. Service or Department number (Reffun)). An order number 1, 2, 3 ... indicates which function(s) will need to **approve** next, assuming no changes to the **approver** list has occurred. Several functions can have the same order number.

**Approver** type (Apptyp) indicates if the function is in the list as an Authorizer (A) or as a Reviewer (R). The Authorizer actions could be "Authorize" or "Reject"; while a reviewer should "**Approve**" or "Disapprove".

An index indicates if the function is mandatory or not on the **approver** list. This will be important to make changes in the **approver** list. If the function is not mandatory, any **approver** can suppress it, but if it is mandatory, either it is not possible to suppress it, or another function must replace it.

APPWAIT contains for each document in **approval** process the function(s) which are actually waiting for the document (i.e. next to act).

As APPFUTU, it contains document identification, function identification, **approver** type and mandatory indicator. It contains also the previous **approver's** name, one line personal comments from the previous **approver**, the date and time of previous action.

APPDONE : contains for each document the functions that have already acted on the document, the decisions and identification of the persons who have acted and of the titular if he is different.

As APPFUTU and APPWAIT, it contains document identification, function identification, **approver** type and mandatory indicator. It contains also the decision of the **approver** Y for Authorize or **Approve**; N for Reject or Disapprove,

the identification of the **approver**, name and employee serial number, the identification of the titular of the function, name and employee number, the delegation indicator, the action date and time.

As already mentioned, the **approval** system for the best mode of this invention is made to use predesigned forms. Any user wishing to start a request for **approval** operation, will access a document form (blank) and fill it in. A form includes predefined fields which, when filled-in will be stored into a ... ..identification, i.e. type and reference of document. Each document is assigned a status (one character) which can be :

P : document is in progress in **approval** process

F : document is finalized, **approval** process is over.

S : document has been sent to an operational system

T : document has been transmitted for action (acknowledgment received from operational system).

H... ..his function (e.g. INDI for employee), his employee serial number and his name.

The table also contains the subject of the request submitted to **approval** (document) and the originating date and time.

In addition to the function tables already mentioned the system will use Specific Function Tables storing data to be used for defining the **approval** rules selected by the form user's company. (see image in original document)

These tables allow retrieving a function reference from a parameter. They will be used during **approver** list determination.

The example shows so called Determination Tables useful to determine the manager who is responsible for a documented project.

Function CHARACTERISTIC tables enable determining parameters from the reference of a given type of function. They will be used during **approver** signature validation. In other words, these tables store predefined **approval** rules.

The example shows the amount of expenses authorized based on the considered manager's function and the company's selected rules setting expense thresholds.

From the above considerations, one may understand that any filled-in form (=document) contents is dynamically used by the system to determine the **approval** path when considered in combination with **approval** rules. In addition, the system is designed to enable adding or modifying forms, as needed, in a fairly simple way. Therefore, the various types of... ..the flow chart of Figure 9).

First, the user is recognized for being logged on a virtual machine which the system knows as a "signature" ( **approval**) machine assigned to a registered user.

The FUNCTION table is the main filter to access the documents. Said table is permanently updated to reflect the...used to find documents on which action has been taken already. Documents partially approved are stored in table APPDONE, while those approved by all required **approvers** (or rejected) are stored in an APPHIST or historic table.

To make the system attractive and end-user friendly from an operational standpoint, panels have... ..menu 1 calls PREPARE EXEC. (See Fig. 10 for a high level flow chart of the process for preparing a document to be submitted to **approval**).

This EXEC starts displaying the list of forms available to the user in a "Choose a Form Category" menu. It should be noted that the system is made to process forms of various categories like "purchasing" orders (APPR), "financial" orders (FINA), requests for getting **approval** to tailor a VM Machine to a new user (LOGO), etc... (see image in original document)

If there are too many categories to fit on... ..system checks in this case that the reference exists in the data-base and that the user is either the originator, or one of the **approvers**. Thus filtering access to said existing document.

The layout of the screens which the user is presented with, depends on the original form designer's...to the header screen

- PF3 Repeat the displayed item to create a new one
- PF4 Add a blank item
- PF5 All operations before sending in **approval** (see below)
- PF10 Display next item if it exists
- PF11 Display previous item if it exists

If the user presses PF12, the system shows the... ..image in original document)

From here, the user will be able to :

- PF1 : View the document as it can be viewed later by all the **approvers**



- PF2 : Change the document data
- PF4 : Add free comments which will be viewed to all the **approvers**
- PF5 : Prepare sending in **approval** (see below)
- PF6 : Save the data entered as a draft document (and retrieve it later)
- PF8 : Print the document. A print image of the document... ...self explanatory flow-chart of Figure 10.

Down to this point the process lead to a fully prepared document ready for being submitted to the **approval** process. Therefore, if the user presses PF5 either from the data entry panels, or from the "Process prepared document", the system performs the following operations... ...found, the system displays again the data entry panel where the field bearing an erroneous data is indicated by the cursor. Then, it determines the **approval** path based on functions involved, specific rules assigned for the type of document involved, and document data (see Figure 14).

If an error is found, the message "Unable to determine **approval** process" is displayed and no action is performed.

Finally, it determines for each function of the **approval** process, the acting person at the present time and the titular.

The result is displayed (see Figure 15) to the considered user (originator) as shown...  
...document)

If the user presses PF12, he just returns to "Process the document" Menu.

Otherwise, by depressing PF1 he will be able to change the **approver** list with some controls and restrictions (see below with reference to Figures 15 and 16).

If he does press PF7, the document is created and waits for action of the first function(s) in the **approval** process (see Figures 15 and 16).

Represented in Figure 11 is a flow-chart summarizing the operations achievable on an already filled-in document. The... ...functional key labeled "Open the Mail", assuming SEALING documents are waiting action. The message looks like:

"You have 5 documents awaiting your action in Electronic **Approval** System. Do you wish to process these ? Type Y and press ENTER if you wish". The system then shows first a list of categories of... ...document)

The user can see on this screen the document type and reference, the subject, the originator's name, who was the last previous **approver** and personal comments from this previous **approver**. Also mentioned is the function ...function if the user is a delegate.

By depressing PF1, the user can view all information about the document, i.e. the document itself; the **approver** list with decisions of **approvers** who have already acted on the document; the document originator and **approvers** comments, if any.

Depressing PF4, enables the user adding his own comments to the other **approver's** comments after acting on the document.

PF5 is not available for all **approvers**. This PF Key is only active if the document is originally tailored to authorize the function to add or modify data. In this case, the user is shown data modification panels and can add or modify data in the document. These modifications can affect the **approval** path process.

PF7 must be used to act on the document.

If the user is an Authorizer, he will see the screen below. (see image in original document)

The user can change the **approvers** list by pressing PF7 (seen later).

The user can Authorize (**Approve**) the document with PF1. The system will display a confirmation panel which looks as follows. (see image in original document)

The next screen shows the next **approver** and allows the user to type one line of personal comments for said next **approver's** attention.

If the user is the last **approver**, the confirmation panel is different and looks as follows : (see image in original document)

Authorization means finalization of the document.

An Authorizer can also Reject... ..his decision, indicating that in case of non confirmation, the document will be rejected, an information will be sent to the originator and to each **approver** who has already acted on the document.

The user may also choose PF3 to request additional information from another **approver**.

In this case, the user has to choose an **approver** in the screen below. (see image in original document)

He can choose the originator, an **approver** who has already acted on the document or an **approver** who has not yet seen the document, then he gets a confirmation panel as below. (see image in original document)

If the user confirms, the document is available for the chosen **approver** and the user will get it back again after this **approver** has acted upon said document. If the user is a Reviewer in the **approval** process, he is shown the following screen. (see image in original document)

There will be two different confirmation panels no matter whether the user is the last **approver** or not. Request for additional information operation is quite the same as requested to an Authorizer. It should be remembered that a Reviewer cannot reject a document. He can just disapprove the document, in which case a further Authorizer **approval** is required. The system determines if there is an authorizer in the list of next **approvers**. If there is no Authorizer in the list of next **approvers**, the system asks the user to choose among the Authorizers who have already acted on the document and will send the document back to the... ..Or the document has been rejected by an Authorizer or has been cancelled by the originator and the user is either the Originator or an **Approver** who has already acted on it.

PF4 has just the effect to cancel reference reminder to said documents. The user can always access the document... ..can access documents using several alternatives (see Figure 12):

- PF1 : Lets the user access the documents he has originated and which are currently in the **approval** process.

NB: by user one means the person assigned corresponding function.

- PF2 : Lets the user access the documents he has processed and which are currently in the **approval** process.

- PF3 : Lets the user access all the documents, whatever their status (in progress, finalized, rejected, cancelled...) he has originated between two dates the user ...the status is "In progress", the user is presented with the screen below. (see image in original document)

The user can - view the document (data, **approvers** list and comments).

- copy the document to create a draft

- print the document

If he is the originator, he can also cancel the document. In... ..user is presented the following confirmation panel. (see image in original document)

Pressing PF7 in "Process the document found" menu allows the originator or an **approver** who has already acted on the document to change the **approver** list as he could have done when originating the document or acting upon the document.

When changes are prepared, the system displays the following panels. (see image in original document)

If all changes are correct, the user can press PF1 and the document proceeds with the new **approver** list.

PF12 will cancel all the changes.

If the status of the document is not "In progress", the available actions are different as shown hereafter. (see image in original document)

The user can - view the document (data, **approver** list and comments).

- copy the document to create a draft

- print the document

Follow-up information may also be given when the user presses PF4... ...looks for documents awaiting action and give the references. For obvious security purposes, no further information about document can be obtained.

As already mentioned an **approver** designation, could be delegated from one user to another under predefined conditions. The user can access this part of the system by pressing PF7 on...  
...In this case, the system considers the user as absent. This feature will be used to bar access to the document for instance when an **approver** is acting on it. So, originators and **approvers** will be informed and will be able to react to this situation.

To validate his entries, the user has to press the enter key.

Using... ...the transfer.

Having thus described from a technical as well as functional standpoint, the means involved in this invention, one may therefore fully comprehend the **approval** system operation. **Approval** system operations has however been summarized in Figures 14 through 17, and Function management in Figure 18.

Represented in Figures 14 through 16 are, the means involved in the determination of the list of **approvers**. Obviously, the system has been limited to simple cases to simplify explaining the operation. One assumes the filled-in document (e.g. **Purchase Order**) includes a Project code, a Purchaser code and an Amount of expenses set by the purchase requesting user. The software means uses the project code... ...FaMANAD2) needs be addressed. Also, due to the type of document involving expenses, the logic adds an Investment Responsible (INVT) to the list of required **approvers**. Same applies to Purchaser (PURC). The system loads the information into two separate tables located in the document originator VM user's machine memory designated as FUTU and DONE respectively. The DONE table contains so called "shadow" **approvers** or virtual **approvers** whom the system will enable read-only access to the corresponding document. As already mentioned, once initiating the **approval** process, the originator may amend the list of **approvers** up to a certain extent based on predefined rules. For instance, deletion of a first line manager from the list of **approvers** will trigger automatic insertion of the second line manager, and so on.

Then, the originator sending decision has the effect to unload the first **approver** references from FUTU table into a NEXTWAIT table while the others are loaded into a NEXTFUTU table in the

preset ordered list, both tables NEXT... ..building and updating the corresponding SQL data base tables, i.e. APPFUTU, APPWAIT and APPDONE of the SEALBDA machine.

The SEALING system also controls mailing **approval** requests to designated **approvers** whose action are controlled as represented in Figure 16.

The designated **approver** in NEXTWAIT gets access to the data in the corresponding SQL/DS tables in its own VM machines into FUTU, WAIT and DONE tables. Then NEXTFUTU, NEXTWAIT and NEXT DONE are set from FUTU, WAIT and DONE tables respectively. These tables contents are used by current **approver** to perform and record the following operations:

- **Approver** list modification: the **approver** may amend the list as the originator was able to do.
- Modification to **approval** process: the system controlled by any amendment to the data of document due to current **approver**, amends the **approval** path accordingly, and
- **Approver** validation: compliance test with the **approver's** designation rules is performed.

Tables updating are performed, i.e. once current **approval** is executed, a shift is operated with next **approver** in NEXTFUTU being shifted into NEXTWAIT.

Once forwarded, the current **approval** data are used to update the SQL/DS tables through add, modify or delete operations. They are then available for next **approver's** action and so on.

A fairly high security level has been provided in this invention to limit false **approvals** due either to human error or to voluntary operation. First, one should notice that **approval**, i.e. insertion of a "Y" or "N" flag into a predetermined field characterizing the considered document is inserted into the SQL/DS Table APPDONE otherwise accessible to the user on a Read-Only basis (see MYSIGNAT in Figure 6). Second, said "signature" or **approval** insertion is operated only upon execution of a predetermined SQL command involving a signature validity check. The flow chart of such a signature validation operation... ..The SQL command triggered by the MYSIGNAT order accesses various SQL tables to gather data stored therein to ensure that the user presently trying to **approve** or disapprove is entitled to do so. First APPWAIT table is accessed to ensure that the considered document defined by typdoc/refdoc is waiting therein... ..Table, together with data and time are enabled upon a positive signature validation test result.

As mentioned, the population of users attached to the same **approval** system is managed in an unique SQL table, i.e. the FUNCTION Table, as represented in Figure 18. For each function, i.e. Manager, Purchaser...

**Claims:** ...A1

1. An **approval** system for controlling the processing of a user originated document requiring signature by electronic **approval** by system selected users, in an electronic mailing system including terminals attached to a digital network, virtual machines (VM) including computer

means, memory and software... ..the population of system attached users, and means for generating processing and monitoring electronic documents to be mailed from any terminal to any user, said **approval** system including:

- means for storing and updating function tables wherein each system user's function and address are identified;
- means for storing document forms;
- means for storing predefined **approval** rules based on user function and document forms contents;
- terminal controllable means for selecting, accessing, filling-in, processing and mailing any selected form whose contents is to be subjected to **approval**;
- means sensitive to said mailing for addressing said function tables and, based upon said **approval** rules, for determining the **approval** path among the system attached users; and,
- means sensitive to said **approval** path determination for monitoring the mailing and processing of said filled-in form accordingly.

2. A system according to claim 1 wherein said means for... ..and means for linking said tables together in a tree shaped arrangement.

3. A system according to claim 2 wherein said means for determining the **approval** path include:

- means sensitive to said **approval** rules for reading document data by addressing specific SQL data tables;
- means sensitive to said data read to address said function tables and fetch **approvers** references therefrom;
- means sensitive to said stored **approval** rules for listing said **approvers** references in a predefined sequential order into an **approval** list; and,
- means for storing said **approval** list into a FUTU table.

4. A system according to claim 3 further including means sensitive to read data for generating an additional list of... ..user;

- means sensitive to said stored rules for enabling said originating user to check and amend said FUTU table; and,
- means for initiating the document **approval** process upon completion of said checking.

6. A system according to claim 5 wherein said means for initiating the document **approval** process include:

- means for loading first **approver** in FUTU table into a NEXTWAIT table and loading remaining FUTU table contents into a NEXTFUTU table; and,

- means sensitive to NEXTWAIT contents to mail a predefined message to the corresponding **approver's** VM machine.

7. A system according to claim 6 including:

- means for unloading said DONE table into a NEXTDONE table within VM user's... ..NEXTDONE user's tables into SQL tables APPFUTU, APPWAIT and APPDONE respectively.

8. A system according to claim 7 wherein said means for managing appropriate **approvals** include:

- user's terminal controllable means for requesting access to system SQL tables;
- system controllable means for accessing stored function tables, comparing terminal user's identification to **approver's** as stored into said function tables and, upon positive check, unloading contents of predefined fields of APPFUTU, APPWAIT and APPDONE SQL tables into user... ..and DONE respectively;
- system controllable means sensitive to said user's machine tables contents for displaying preselected data from documents waiting for said user's **approval**;
- user's terminal controllable means for selecting one of said documents whereby said selected document is being displayed to said user; and
- , - terminal controllable means for said user inserting **approval** or disapproval decision (signature) into a predefined document field.

9. A system according to claim 8 wherein said means for inserting **approval/disapproval** user's decision include system controlled signature validation means and, upon correlative positive validity check, means sensitive to said validation means for writing user...

**Claims: ...B1**

1. A system for controlling the processing and routing of a user originated document requiring electronic **approval** by selected system users, in an electronic mailing system including terminals (T1, T2, T3) attached to a digital network, user's Virtual Machines (VM) including... ..system attached users, and software controlled computer means for generating, processing and monitoring electronic documents to be mailed from any terminal to any user for **approval**, said **approval** system including means for scheduling **approval** tasks for users and means for defining a path or route specification for any document to be approved by various users, said means for scheduling **approval** and specifying a path being characterized in that it includes :

first storage means (SEALDBA Machine; Function Tables) and terminal controllable means for storing and updating... ..Structured Query Language (SQL) form, and software operated computer means for linking said tables together in a tree-shaped arrangement ;

third storage means (SEALDBA Machine **Approval** Tables) for storing predefined **approval** rules based on user's function and document forms contents ;

terminal controllable means (User's Virtual Machine) for selecting, accessing, filling-in, processing and mailing any selected form whose contents is to be subjected to **approval**;

first software controlled computer means sensitive to said mailing for addressing said function tables and, based upon said **approval** rules, for **dynamically** determining the **approval** path or route among the system attached users, said means for determining the **approval** route including :

second software controlled computer means sensitive to said **approval** rules for reading document data by addressing specific SQL data tables;

third software controlled computer means (FQMANAD1, FaMANAD2) sensitive to said read data to address said function tables and fetch **approvers'** references therefrom ;

fourth software controlled computer means sensitive to said stored **approval** rules for listing said **approvers'** references in a predefined sequential order into an **approval** list ; and,

fourth storage means for storing said **approval** list into a table (FUTU) whereby said **approval** path is determined and used for monitoring the mailing and processing of said filled-in form accordingly.

2. A system according to claim 1 further... ..said stored rules for enabling said originating user to check and amend said FUTU table ; and,

seventh software controlled computer means for initiating the document **approval** process upon completion of said checking.

4. A system according to claim 3 wherein said means for initiating the document **approval** process include :

eighth software controlled computer means for loading first **approver** in the FUTU table into a prestored NEXTWAIT table and loading remaining FUTU table contents into a prestored NEXTFUTU table ; and,

ninth software controlled computer means sensitive to NEXTWAIT table contents to mail a predefined message to the corresponding **approver's** VM-machine.

5. A system according to claim 4 including :

tenth software controlled computer means for unloading said DONE table into a prestored NEXTDONE ... ..user's tables into prestored SQL tables APPFUTU, APPWAIT and APPDONE respectively.

6. A system according to claim 5 wherein said means for managing appropriate **approvals** include :



user's terminal controllable means for requesting access to system SQL tables ;

system controllable means for accessing stored function tables, comparing terminal user's identification to **approver's** as stored into said function tables and, upon positive check, unloading contents of predefined fields of APPFUTU, APPWAIT and APPDONE SQL tables into user... ..and DONE respectively ;

system controllable means sensitive to said user's machine tables contents for displaying preselected data from documents waiting for said user's **approval** ;

user's terminal controllable means for selecting one of said documents whereby said selected document is being displayed to said user ; and

terminal controllable means for said user inserting **approval** or disapproval decision (signature) into a predefined document field.

7. A system according to claim 6 wherein said means for inserting user's decision include system controlled decision (**approval**) validation means and, upon correlative positive validity check, means sensitive to said validation means for writing user's decision into SQL table APPDONE otherwise accessible on a read-only basis.

8. A system according to claim 7 wherein said **approval** validation means include software controlled computer means for fetching a system prestored SQL command and means sensitive to said command for triggering said **approval** validity check.

9. A system according to claim 8 wherein said **approval** validation means include :

software controlled computer means for addressing the APPWAIT system table and checking presence of the considered document therein; and,

software controlled computer...

8/3K/4 (Item 1 from file: 349)  
DIALOG(R)File 349: PCT FULLTEXT  
(c) 2011 WIPO/Thomson. All rights reserved.

00935095

**DYNAMIC PAYMENT CARDS AND RELATED MANAGEMENT SYSTEMS AND ASSOCIATED METHODS**  
CARTES DE PAIEMENT DYNAMIQUES, SYSTEMES DE GESTION ASSOCIES ET PROCEDES ASSOCIES

**Patent Applicant/Patent Assignee:**

- WORKS OPERATING COMPANY**

6801 N. Capital of Texas Hwy., Building 1, Austin, TX 78731; US; US(Residence);  
US(Nationality)

**Inventor(s):**

- PRAISNER C Todd**  
74 St. Stephens School Road, Austin, TX 78746; US
- HOLLAND IV James R**  
11327 Alhambra, Austin, TX 78759; US
- KIPP JR Roy H**  
16322 Spotted Eagle Drive, Leander, TX 78641; US
- BALBACH Melissa T**  
645 Sycamore Lane, Glencoe, IL 600022; US
- LEISEROWITZ William R**  
11019 Galleria Cove, Austin, TX 78759; US

**Legal Representative:**

- PETERMAN Brian W (agent)**

O'Keefe, Egan & Peterman, 1101 Capital of Texas Highway South, Suite C-200, Austin,  
TX 78746; US

	Country	Number	Kind	Date
Patent	WO	200269290	A2-A3	20020906
Application	WO	2001US51418		20011019
Priorities	US	2000242493		20001023
	US	2000717728		20001121
	US	2001276819		20010316

**Designated States:** (Protection type is "Patent" unless otherwise stated - for applications prior to 2004)

AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG,  
BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ,  
DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD,  
GE, GH, GM, HR, HU, ID, IL, IN, IS, JP,  
KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT,  
LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ,  
NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG,  
SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG,  
UZ, VN, YU, ZA, ZW

[EP] AT; BE; CH; CY; DE; DK; ES; FI; FR; GB;  
GR; IE; IT; LU; MC; NL; PT; SE; TR;

[OA] BF; BJ; CF; CG; CI; CM; GA; GN; GQ; GW;  
ML; MR; NE; SN; TD; TG;

[AP] GH; GM; KE; LS; MW; MZ; SD; SL; SZ; TZ;  
UG; ZW;

[EA] AM; AZ; BY; KG; KZ; MD; RU; TJ; TM;

**Language** Publication Language: English

Filing Language: English

Fulltext word count: 33407

### English Abstract:

...management system can automatically interface with card processor systems to dynamically modify these card control settings, and a purchasing management system or other request and **approval** workflow engine can provide an interface between a company and the dynamic card management system. More generally, an advantageous solution for purchasing management is disclosed that utilizes **dynamically** or actively managed **approval** parameters to help control transaction authorization determinations associated with purchasing mechanisms. These **dynamic approval** parameters can be generated and/or managed through the application of configurable company purchasing policies and rules, and these **dynamic approval** parameters can be stored, for example, by a processing system that makes authorization determinations when transactions are initiated using the purchasing mechanisms.

### Detailed Description:

...made as company or organizational expenses, such as purchases of office products, office supplies, services, and marketing materials, are often handled through purchase request and **approval** processes with associated payment procedures. These procedures, however, are typically inefficient. In addition to purchase request and **approval** processes, reimbursement procedures are also used for corporate expenditures. For example, individual travel and entertainment expenditures are often handled through reimbursement procedures where employees must... ..being purchased, it is desirable for companies to have the ability to manage the purchasing of those products and services through one common purchasing and **approval** process.

Typical purchasing management solutions have been business-to-business electronic commerce applications that automate product procurement. These systems, however, are often simple Web-based... ..typically very reluctant to issue company credit or debit cards, thereby giving employees the ability to create company debt or expend company funds without pre-**approval**. If employee accounts are used, the transaction is treated as a reimbursement transaction.

Stored value cards are essentially pre-paid credit cards with pre-determined... ..help company

payment needs. Purchasing cards or "I" cards are essentially company credit cards with a few additional control features that allow companies to **pre-approve** cards using a few static limitations, such as pre-approved vendors, preapproved transaction amount limits, and pre-approved overall credit limits. Once set up, these... ..card, and specific company constraints at both the master account and individual card levels. Based on traditional card processing capabilities, the types of constraint and **approval** parameters that can be statically configured by the bank for purchasing cards include the following example control features as shown in TABLE 1.

TABLE 1...management system can automatically interface with card processor systems to dynamically modify these card control settings, and a purchasing management system or other request and **approval** In one embodiment, the present invention is a method for dynamically managing payment card control settings, including receiving a purchase request from a requestor within an entity, processing the purchase request with respect to purchase policies for the entity, **approving** the purchase request if the purchase policies are satisfied, reviewing control settings for a payment card associated with the approved purchase request, and adjusting the... ..management rules that reside on one or more server systems, and the customizable purchasing management rules can include an ability to configure organization structures and **approval** chains. Further, the method can include notifying an **approver** of a purchase request, if some action is required from the **approver** for the purchase request to be approved, and allowing the **approver** to take the required action through a network accessible **approval** mechanism. This network may include the Internet. Still further, the method may include associating a payment card with an element within the entity, which can... ..detailed respects, the method may include a card processor that stores the payment card control settings. The method may further include generating a set of **approval** parameters for the approved purchase request and comparing the **approval** parameters with the control settings to determine what adjustments to make so that the purchase may be made with the payment card. In addition, the control settings may correspond to the **approval** parameters. The method may also include the card processor acting to compare parameters of an attempted purchase transaction with the control settings and to authorize...certain purchasing capabilities with respect to a payment mechanism, communicating from the one or more server systems to a processing system to obtain information representing **approval** parameters associated with the payment mechanism, and sending from the one or more server systems to the processing system adjustment instructions to adjust the **approval** parameters for the payment mechanism so that the purchasing capabilities are available, wherein the processing system is a system that processes transactions initiated using the payment mechanism based upon **approval** parameters associated with the payment mechanism and stored by the processing system. In addition, the request can include an approved purchase request related to one... ..generated by receiving a purchase request from a requestor within the entity, processing the purchase request with respect to purchase policies for the entity, and **approving** the purchase request if the purchase policies are satisfied, and generating the request based upon the purchase request. Still further, the one or more server... ..detailed respects, the method may include a request that includes a request to provide pre-approved purchasing authority for the payment mechanism resulting in the **approval** parameters being adjusted to provide this pre-approved purchasing authority. The one or more server systems can also receive transaction data associated with a transaction ... ..can be further processed with respect to purchase policies for the entity and can be approved if the purchase policies are satisfied. Still further, the **approval**

parameters can be adjusted to restore the pre-approved purchasing authority based upon an occurrence of one or more selected events after completion of the... ..be associated with each payment mechanism. The payment mechanism can include a payment card, the processing system can include a payment card processing system, the **approval** parameters can include control settings for the payment card, and the one or more server systems can include a purchasing management system.

In still further... ..the dynamic payment identifier associated with it. The request can include a purchase request that has been approved based upon entity purchasing policies, wherein the **approval** parameters can include a set of **dynamic approval** parameters associated with the approved purchase request, and wherein the processing system stores the set of **dynamic approval** parameters for the **approved** purchase request. In addition, the method can include receiving a plurality of purchase requests associated with a particular dynamic payment identifier and further include communicating a set of **dynamic approval** parameters for each purchase request from the one or more server systems to the processing system, with each set of **dynamic approval** parameters being stored by the processing system. Still further, the method may include the card processing system comparing - 7 parameters of an attempted purchase transaction made using the dynamic payment identifier with the sets of **dynamic approval** parameters stored for that **dynamic** payment identifier and authorizing the purchase transaction if the parameters match at least one set of **dynamic approval** parameters.

In another embodiment, the present invention is a system for dynamically managing payment card control settings including one or more systems configured to receive a purchase request from a requestor within an entity, to process the purchase request with respect to purchase policies for the entity, to **approve** the purchase request if the purchase policies are satisfied, to review control settings for a payment card associated with the approved purchase request, and to... ..rules that reside on the one or more server systems, and the customizable purchasing management rules can include an ability to configure organization structures and **approval** chains. In addition, the server systems can be configured to notify an **approver** of a purchase request, if some action is required from the **approver** for the purchase request to be approved, and further comprising a network accessible **approval** mechanism through which the **approver** can take the required action. And this network can include the Internet. Still further, the payment card can be associated with an element within the... ..card processor that stores the payment card control settings. And the one or more server systems can be further configured to generate a set of **approval** parameters for the approved purchase request and to compare the **approval** parameters with the control settings to determine what adjustments to make so that the purchase may be made with the payment card. In addition, the control settings can correspond to the **approval** parameters.

The card processor can also be configured to compare parameters of an attempted purchase transaction with the control settings and to authorize the purchase... ..within an entity to make available certain purchasing capabilities with respect to a payment mechanism; to communicate with a processing system to obtain information representing **approval** parameters associated with the payment mechanism that are stored and used by the processing system to process transactions initiated using the payment mechanism based upon the **approval** parameters associated with the payment mechanism; and to send to the processing system adjustment

instructions to adjust the **approval** parameters for the payment mechanism so that the purchasing capabilities are available. More specifically, the request can include an approved purchase request related to one... ..to receive a purchase request from a requestor within the entity, to process the purchase request with respect to purchase policies for the entity, to **approve** the purchase request if the purchase policies are satisfied, and to generate the request based upon the purchase request. The one or more server systems... ..more detailed respects, the server-based system can include requests that include requests to provide pre-approved purchasing authority for the payment mechanism, and the **approval** parameters can be adjusted to provide this pre-approved purchasing authority. Also, the one or more server systems can be further configured to receive... ..transaction data. The server systems can also be configured to process the synthesized purchase request with respect to purchase policies for the entity and to **approve** the synthesized purchase request if the purchase policies are satisfied. Still further, the server systems can be configured to send adjustment instructions to adjust the **approval** parameters to restore the pre-approved purchasing

authority based upon an occurrence of one or more selected events after completion of the transaction. Still... ..detailed respects, the server-based system can include a payment mechanism that includes payment cards, a processing system that includes a payment card processing system, **approval** parameters that include control settings for the payment card, and one or more server systems that include a purchasing management system. In addition, the server...the request received by the server system or systems can include a purchase request that has been approved based upon entity purchasing policies, wherein the **approval** parameters include a set of **dynamic approval** parameters associated with the approved purchase request, and wherein the processing system is configured to store the set of **dynamic approval** parameters for the **approved** purchase request. Also, the server systems can further be configured to receive a plurality of purchase requests associated with a particular dynamic payment identifier and to communicate a set of **dynamic approval** parameters for each purchase request from the one or more server systems to the processing system, with each set of **dynamic approval** parameters being stored by the processing system. Still further, the card processing system can be configured to compare parameters of an attempted purchase transaction made using the dynamic payment identifier with the sets of **dynamic approval** parameters stored for that **dynamic** payment identifier and to authorize the purchase transaction if the parameters match at least one set of **dynamic approval** parameters.

#### Description of the Drawings

It is noted that the appended drawings illustrate only exemplary embodiments of the invention and are, therefore, not to be... ..a purchasing management environment utilizing dynamic payment cards, according to the present invention.

FIG. 2 is a flow diagram for purchase request processing that utilizes **dynamic** payment identifiers and **approval** parameters, according to the present invention.

FIG. 3 is a block diagram representing various potential sources for purchase requests within a purchasing management environment, according to the present invention. FIG. 4 is a block diagram for a dynamic purchasing card management environment, according to the present invention.

FIG. 5B is a block diagram for example forward flow **approval** processing and reverse flow **approval** processing for a purchasing card management environment, according to the present invention.

FIG. 6A is a flow diagram for dynamic payment card processing where the... including spending controls, request management, order management and financial management.

FIGS. 9A, 9B, 9C and 9D are swimlane diagrams for account setup and card issuance, **approval** required transactions, discretionary transactions, and logical matching.

FIG. 10A is a block diagram for a secure proxy environment to facilitate secure communications between the purchasing... dynamically adjusted and modified based upon purchasing needs. Using the purchasing management system of the present invention, a company may set up purchasing policies and **approval** rules and then have those policies and rules - 12 harmonized with dynamic payment cards provided to its employees. When the purchasing management system determines that... thereby providing dynamic management of card control settings based upon purchasing needs. When a transaction is initiated, the card processor can then look to these **dynamic** control features to provide **approval** information to the merchant. Transaction data can then be provided back from the card processor to the purchasing management system for matching, reconciliation and accounting... card processing according to the present invention are further described below.

More generally, the present invention achieves an advantageous solution for purchasing management by utilizing **dynamic approval** parameters, such as **approval** parameters that are **dynamically** or actively managed based upon purchasing needs, to help control transaction 5 authorization determinations associated with purchasing mechanisms. These **dynamic approval** parameters can be generated and/or managed through the application of configurable company purchasing policies and rules, and these **dynamic approval** parameters can be stored, for example, by a processing system that makes authorization determinations when transactions are initiated using the purchasing mechanisms. Although a wide... payment cards according to these card control settings.

TABLE 1 above provides an example of such control settings. The present invention can provide and manage **dynamic approval** parameters through the **dynamic** management of these payment card control settings that are stored and utilized by traditional card processors. As applied to such existing payment card infrastructure, the... to the dynamic management of card control settings traditionally stored by card processors, the dynamic purchasing management system provided by the present invention further contemplates **approval** parameters that are more robust than these traditional control settings.

For example, transaction-specific **approval** parameters could be stored by card processors, and these settings could also be dynamically managed by the purchasing management system. These more detailed **dynamic payment approval** parameters could be adopted and utilized, for example, such that sets of **approval** parameters are linked to individual purchase requests and associated transactions.

**Dynamic approval** parameters for any particular purchase request can be produced, for example, after applying the company purchase policies to those purchase requests within a purchasing management system. The purchasing management system can then compare these **approval** parameters with the control settings stored by the card processor for the payment card to determine if the card settings need to be adjusted to allow authorization of a transaction associated with the approved purchase request.

15 By utilizing dynamically managed payment card control settings or other **dynamic approval** parameters linked to purchase requests, as opposed to general, non-managed and static **approval** limitations available with existing purchasing card programs, the systems and methods of the present invention allow for the efficient management and control of company purchases...  
...purchase policies. This advantageous purchase processing utilizes the dynamic payment identifiers, such as dynamic payment cards, to provide a mechanism for identifying users so that **dynamic approval** parameters associated with purchase requests from a user or an element (e.g., people, equipment, vehicles, buildings, or any other desired person, place or thing... ..a payment card as a primary mechanism for managing spending and procurement in business operations, while still maintaining active control over those purchases. The static **approval** mode of traditional purchasing cards simply does not support the standard purchase **approval** process in most businesses, where managers review requests and then **approve** these specific requests for purchase.

Many companies are not willing to use such traditional purchasing cards for the majority of their expenditures due to the... reimbursement or check request procedures.

As described herein, the dynamic payment card platform of the current invention resolves this situation by providing significant management and **approval** capabilities that exceed those available with traditional purchasing card programs. Advantageously, with the present invention, review of purchase requests may occur before purchase execution, enabling the dynamic payment card to actively support standard company purchasing processes. **Approval** processing then allows for robust exception handling procedures and enables the use of purchase requests that trigger rules for desired **approval** routing. And, once approved, **approval** requirements for a particular purchase request may be tied or associated with a particular dynamic payment card so that card transactions may be processed within a credit card processing network, while still allowing for **approval** and policy control by the business. In addition, the processing mechanism for the dynamic payment card - 15 may utilize existing credit card networks, for example...  
...provides an example block diagram for a dynamic payment card management system environment. FIGS. 2 and 4 provide example flow diagrams for transactions that utilize **dynamic** payment cards and **dynamic approval** parameters. FIG. 3 provides a block diagram of example sources for purchase requests that are to be processed by the purchasing management system. FIGS. 8A... ..parameters among the various entities within the payment card infrastructure.

It is noted that the dynamic management of existing payment card control settings as the **dynamic approval** parameters is likely to be the embodiment of the present invention initially implemented because payment cards are likely the purchasing mechanism to be initially adopted... ..for dynamic management of existing payment card control settings. FIG. 2 and FIG. 4 will next be discussed. These drawings are directed more generally to **approval** parameters



being applied based upon purchase requests. FIG. 3 is then discussed. Finally, a series of example company transactions are provided along with a discussion... ..to be utilized. It is noted that although FIG. 1 depicts a purchasing management system 100 that is a server-based network-accessible system, the **dynamic approval** parameters and payment card of the present invention may be utilized with client-based systems, server-based systems, or a combination of both. It is... ..represent the various employees that utilize the purchasing management system 100. These users can in one sense be thought of as requesters who request purchases, **approvers** who **approve** requests, and administrators who manage the purchase policies and rules. Additionally, logical organization can be accomplished through groups of elements, where groups can be organized...unique numbers may be utilized, such as social security numbers. These unique numbers initiated transactions.

The purchasing management system 100 includes purchase request block 116, **approval** processing block 118, **dynamic** purchase processing block 120, and transaction reconciliation and reporting block 122. The purchase request block 116 represents subsystems within the purchasing management system 100 that ... ..requests by users 106A, 106B ...

106C@ 108A@ 108B ... 108C, and users 1 1 OA, 1 1 OB ... 1 1 OC within clients 104A, 104B ... 104C.

**Approval** processing block 118 represents subsystems that allow for the processing of purchase request utilizing policies and rules that may be configured, selected or otherwise put... ..block 120 represents subsystems that handle purchase requests utilizing dynamic payment cards. This block communicates with the dynamic @ayment card processing system 130 and provides **purchase order** information to the transaction reconciliation and reporting block 122. This block 122 represents subsystems that reconcile approved purchase requests with processed transaction information received from... ..users and/or - 18 purchasing managers a wide variety of network accessible and configurable controls for managing purchases. These customizable management features include automatic purchase **approval**, manual purchase **approval** through **approval** queues, automatic order placing after purchase **approval**, and post-purchase management features.

These policies and rules for **approval** processing configuration within purchasing management system 100 can be implemented as desired to cover the expected and/or unusual or unexpected purchasing **approval** needs of the company. For example, spending rules and policies that may be defined and configured to influence routing and/or **approval** requirements can include features such as cumulative spending/budget based rules, dollar value rules, spend category/expense type rules, payment type/method rules, supplier or type... ..and when products will be shipped and received. In addition, such spending rules operating on the purchasing management system may be configured to trigger manual **approval**, automated **approval**, automated **approval** with notification (for example, where a request is automatically approved using the spending rules logic, but one or more elements and/or groups are sent a notification of the **approval** which includes the reason(s) a notification of **approval** is being received), or any other type of trigger a reasonable user might desire. Furthermore, the spending rules utilized can be applied to groups (departments... ..any element(s) and/or group(s) or any combination of element(s) and/or group(s)).

The dynamic payment card processing system 130 includes **approval** parameters database block 132 and transaction processing block 134. The **approval** parameters database block 132 represents subsystems that store **approval** parameters that are sent from the dynamic purchase processing block 120.. These **approval** parameters, for example, are stored and/or managed with respect to each specific purchase request and are, therefore, dynamic with respect to initiated transactions. Thus, when a user initiates a transaction, vendor systems 140 communicate with the transaction processing block 134. If the transaction falls outside of the **approval** parameters, the transaction processing block 134 rejects the transaction and notifies the vendor systems 140 of this rejection. If the transaction falls within the **approval** parameters, the transaction processing block 134 approves the transactions and notifies the vendor systems 140 of this **approval**. The transaction block 134 then sends transaction details to the transaction reconciliation and reporting block 122. It is noted that the dynamic payment card processing... ..existing credit card transaction systems, such as the VisaNet credit card processing network, with the additional ability to dynamically store or manage a set of **approval** parameters per approved purchase requests. As indicated above, traditional purchasing card processing only stores a single, static set of limitations for any given purchasing card.

By storing detailed **approval** parameters for each requested transaction, the present invention can provide a **dynamic**, transaction-based pre-**approval** system for each transaction. In this way, each transaction using the dynamic payment card may be dynamically pre-validated by a set of **approval** parameters, such as merchant, transaction amount, timeframe or any other desired parameter. As disclosed herein, the **dynamic** payment card and **approval** parameters of the present invention allow for control and management of purchasing regardless of the product or service being purchased.

Thus, the purchasing management system of the present invention allows for purchasing management of any given purchasing need. Upon **approval** of a purchase request, the server systems operating the purchasing management system can initiate a transaction through an agent of a credit card network.

That transaction can authorize the **approved** expenditure for a specific **dynamic** payment card. Once the credit card network sees a customer transaction against that dynamic payment card, the purchasing management system will be notified by the... ..the purchasing management system 100 and are then provided by the purchasing management system 100 to the card processing system 130 and stored in the **approval** parameters database 132. These parameters allow the card holder to have pre-approved spending authority without the requirement of first generating a purchase request that... ..be given an opportunity to provide a user justification or other information, and the justified or the unjustified synthesized purchase request may be subjected to **approval** processing, if desired. As discussed more fully with respect to FIGS. 5A-B, 6A-B and 7A-13, the paths starting in the embodiment of FIG. 1 with blocks 116 and 124 provide example processing paths that allow for forward flow **approval** processing and reverse flow **approval** processing.

As discussed above, and in more detail below, the present invention can be applied to dynamically manage payment cards control settings stored by card processors to provide the

**dynamic approval** parameters. This **dynamic** management of control settings can be based upon company purchasing policies without requiring the company to employ a card processing specialist. In other Looking now... interfaces with the purchasing management (PM) system 100. As discussed above, these interactions 510 allow for the company 104 to define purchasing policies and associated **approval** processing rules within the PM system 100. In addition, the interactions 510 allow the user roles and card settings to be selected and managed. It is also through these interactions 510 that users 106A, 106B ... 106C can make purchase requests, that these requests can be processed, and that **approval** responses can be provided. Based upon these rules, roles and **approval** processing, the PM system 100 interfaces with the card processing system 130 to dynamically manage the settings ... two different paths through which a dynamic payment card can be managed and utilized. In the example shown, processing flow 550 includes a forward flow **approval** processing path 554 and reverse flow **approval** processing path 556 that may be utilized depending upon how the user 552 uses the dynamic payment card. It is noted that these two paths... be taken as being all of the processing paths that could be implemented using the dynamic payment cards of the present invention.

- 22 If the **dynamic** payment card has pre-**approved** spending authority associated with it, the user may proceed down line 564 to begin the reverse flow **approval** processing path 556. Block 566 represents pre-set spending allowances that have been set up for the dynamic payment card being used by the user 552. The user 552 then uses the card to make a purchase in block 568 where card processing occurs, authorizing the transaction if the **approval** parameters for the card are satisfied.

Based upon information available from the card processor 130 about the purchase, the PM system 100 can synthesize post ... or requested information, for the purchase made in block 568 and provides for post-transaction processing of the purchase, for example, through the company's **approval** processing rules, if desired.

Thus, this request synthesis advantageously allows purchases made using pre-approved spending authority to be input back into the PM system 100 for post-transaction **approval** based on the company's purchasing **approval** guidelines and purchase reconciliation. It is noted that if the user 552 proceeds down the reverse flow path 556 without having a card with pre... made under general pre-allowed card constraints to be subjected to post-transaction processing and reporting by the purchasing management system and to post-transaction **approval** routing, if desired. As discussed with respect to the example below, dynamic payment cards may be set-up with some level of approved spending authority without requiring the user first to go through the purchase request **approval** process. If a user makes a transaction using such pre-allowed spending authority, that purchase will typically not have a purchase request or **purchase order** associated with it. Using data provided from the card processing system 130, the purchasing management system 100 can synthesize a purchase request and/or a **purchase order** for this transaction. This synthesized purchase request/order can then be used by a company's accounting system so that the transaction can be reconciled... disapproved thereby requiring reimbursement by the user to the company.

23 The user 552 may also proceed down line 558 to begin the forward flow **approval** processing path 554. In block 559, the user 552 initiates a purchase request in the PM system 100 and the

request is subjected to pre-transaction **approval** processing, as discussed above. Assuming **approval** is granted, the PM system 100 in block 560 then automatically and dynamically manages the card settings so that these settings will allow for the... ..the embodiment of FIG. 5A, the PM system 100 interfaces with the card processing system 130 to determine the current settings for the card's' **approval** parameters and to create or modify those settings so that they will allow the transaction that is the subject of the approved purchase request. In block 562, the user 552 makes the purchase and the purchase is authorized if it is allowed by the card settings and **approval** parameters. As discussed in more detail with respect to FIGS. 9A-D, summary transaction information for a company's cards may be periodically received from...least two basic roles within the dynamic payment card management system 100. These are the Requestor role, which represents a user who typically must obtain **approval** before being authorized to make purchases for the corporation, and the Buyer role, which represents a user who typically has some level of pre-approved... ..and least powerful role in the I O direct or indirect spending equation. Requesters do not generally have any spending authority unless they have prior **approval** from the company. Prior **approval** for the Requester may arise from a variety of circumstances, but will often arise from one of two sources. The first source of preapproval is from an approved purchase request via the company's purchasing process, perhaps an approved "check request" or **purchase order** (PO). A second source or pre-**approval** is a budget that the company has granted to the Requester for specific types of purchases. That budget may be further limited by factors such... ..company if those expenses are determined to be reimbursable within the company's policies. It is noted that in addition to these two typical pre-**approval** circumstances, a company may want to pre- **approve** spending authority for any of a variety of reasons.

The Buyer role can be viewed as a purchasing role in the company covering one or... ..procurement of all technology products for a company. The Office Manager is another common type of Buyer, generally overseeing the companies office supplies purchases by **approving**, aggregating, and placing orders with office product suppliers. Buyers often act as the gatekeeper to suppliers in their area as Requester purchase requests are approved...on the technology Buyer example, the purchase of a new and expensive 0 software package with a recurring annual maintenance fee would likely require pre-**approval**. Most companies have this hybrid approach, allowing Buyer discretion and post-purchase reconciliation on certain purchases and requiring pre-**approval** on others. From an accounting perspective, each Buyer purchase may be reconciled against a centralized company budget for the designated category of spend or against... ..As stated above, the dynamic payment card of the present invention can have any of a variety of settings associated with it. For example, the **approval** parameters for each approved transaction may be stored by the card processor. The level of detail for these **approval** parameters depends upon the capabilities of the card processor 130 and upon the desires of the user. For example, as discussed above, existing purchasing cards... ..amounts, and diversion accounts. These control features may be utilized to constrain how the card may be used to make purchases. Depending upon the desired **approval** setting's and the control features utilized by the card processor, the dynamic payment card 5 management system of the present invention can provide from...diagram for example interactions for payment card account setup and card issuance. FIG. 9B provides a swimlane diagram for example interactions for a transaction where **approval** is first required, for example, as with respect to forward flow **approval** processing path 554 of FIG. 513. FIG. 9C provides a swimlane diagram for example

interactions for a transaction including discretionary spending authority, for example, as with respect to reverse flow **approval** processing path 556 of FIG. 5B. And FIG. 9D provides a swimlane diagram for example interactions for adding order identifiers to facilitate logical inatching of. ... represents an interaction between the card administrator 104 and the cardholder 106 to deliver the card to the cardholder.

FIG. 9B provides example interactions for **approval** required transaction processes 930. The entities listed at the top of the swimlane diagram are the PM system 100, the card administrator or customer 104, the cardholder 106, **approver(s)** 902, accountant 904, merchant 140 and card processor 130. In FIG. 913, the processes 930 are separated into four activities, namely requesting approval process 929, update card settings 931, purchasing process 932 and transactional data processing 933.

Within the requesting/**approval** process activity 929, interaction 934 represents an interaction between the cardholder 106 and the PM system 100 to create a purchase request. Interaction 935 represents an interaction between the PM system 100 and **approver(s)** 902 to provide notification that **approval** action is needed, if **approval** is required. And interaction 936 represents an interaction between **approver(s)** 902 and the PM system 100 to indicate **approval** of the request.

Within the update card settings activity 931, interaction 937 represents an interaction between the PM system 100 and the card processor 130. ... settings and available spend authority. An @ interaction 939 represents an interaction between the PM system 100 and the cardholder 106 to provide notification of request **approval**.

Within the purchasing process activity 932, interaction 940 represents an interaction between the cardholder 106 and the PM system 100 to provide a verification of ... to indicate that a transaction notification trigger (or triggers) has been met.

Similarly, interaction 947 represents an interaction between the PM system 100 and the **approver(s)** 902 that a transaction notification trigger (or triggers) has been met. Triggers can be set, for example, to activate based upon the occurrence of ... FIG. 9B. In addition, in FIG. 9C, the processes 960 are separated into the same four activities as in FIG. 9B. Within the - 37 requesting/**approval** process activity 929, interaction 961 represents an interaction between the card administrator 104 and the PM system 100 to assign discretionary or pre-approved spend or purchasing authority, for example, with initial card settings. These pre-approved purchasing capabilities allow for discretionary spending activity without the requirement of purchase request **approval** processing and subsequent allocation of available spending capabilities with respect to a payment card. Within the other activities 931, 932 and 933, the interactions are. ... system 100 or through a combination of manual and automated activities. For example, a payment card is used to make two purchases through purchase request **approval** procedures as shown in FIG. 913, resulting in Approved Purchase Request 1 (APR1), approved Purchase Request 2 (APR2), Transaction 1 (T1) for \$500, and Transaction... be a merchant identifier or code, a transaction type identifier or code, a one-bit flag that identifies the type of purchase transaction (e.g., **approval** required or pre-approved spending authority), or any other desired identification information.

Within the requesting/ **approval** process activity 929, interaction 934 represents an interaction between the cardholder 106 and the request/order interface 906 of the PM system 100 to create a purchase request. Interaction 936 represents an interaction between **approver(s)** 902 and the request/order interface 906 of the PM system 100 to indicate **approval** of the request. And interaction 981 represents an additional interaction, which may be part of the interaction 936, between **approver(s)** 902 and the... ..provided in some other manner or by another entity or participant, for example, the Order ID could be provided by a cardholder, a requestor, an **approver**, an administrator, a supplier, an accountant or a buyer, and it could be provided automatically by the system.

Within the update card settings activity 931... ..And interaction 939 - 40 represents an interaction between the request/order interface 906 of the P system 00 cardholder 106 to provide notification of request **approval**.

Within the purchasing process activity 932, interaction 940 represents an interaction between the cardholder 106 and the request/order interface 906 of the PM system ...set of spending rules.

Organizational rules and policies can include activities such as creating hierarchies of groups, 1 0 assigning users to these groups, establishing **approval** routing paths through the groups or persons within these groups, creating relationships between groups with related spending limits and budget constraints, or creating any other desired organizational related rule. Financial rules can include activities such as assigning **approver**, requestor and administrator roles and authorities to persons within the organization, implementing corporate-wide or group-level spending policy rules, 1 5 implementing other spending... ..other desired supplier related rule. As part of the spending control activities 802 and related configurations, for example, companies can configure authority levels in the **approval** and spending processes, can set up corporate spending policies, and can configure group structures by organization, project or spending categories. In short, through spending control... ..include, for example, configurable purchase policies that allow companies to identify and establish various purchase rules and policies that automatically determine what is required for **approval** of various purchase requests. For example, such purchase policies may include the ability to determine what purchases may be automatically approved, as well as an indication of what purchases require manual **approval** before being purchased. In addition, the degree of manual **approval** may be selected, as desired, from simply **approval** of an amount to be spent to ever increasing details concerning the vendor or other details of the purchase.

- 42 In addition, in one embodiment... ..other groups. Spending limits may be established for groups, elements, or combinations of both, individually or in aggregate. Elements can be defined as requesters or **approvers** or any other desired designation. The manager may then define detailed options for purchase **approval** including rules that determine when given purchases require manual **approval**. Advantageously, this networkaccessible management system may be implemented with traditional network browsers without requiring special client-side software.

Example **approval** management features for purchasing management systems are described in more detail in U.S. Patent Application SN 09/409,316, entitled "Method and System for... ..helps describe example processes that can be a part of the request management activities 804.

In this embodiment, there are a Requester (Billy Houston), an **Approver** (John Smith) and the potential for additional **approver(s)**. As noted at the bottom of FIG. 8B, the request management activities 804 are primarily internal corporate activities, although external requestors or **approvers** could be included, if desired.

In block 812, purchase requests by the Requester can be created. These requests can be saved in block 810, and... ..822 that notification, such as through an e-mail notification or through another desired notification mechanism, of the purchase request can be sent to an **Approver** should **approval** be required by the rules and policies set up through the spending controls activities 802. In block 824, the purchase requests are reviewed. If rejected... ..block 818. If the purchase request is approved in block 824, flow can proceed to block 826 where it 43 can be determined whether additional **approval** is required. If the answer is "YES," flow can proceed to an additional **approval** cycle as indicated in FIG. 8B. If the answer is "NO," flow can continue to order management activities 806 as represented by element 828.

Purchase request processing may also utilize **approval** queues. Once items are sent to the **approval** queues, a notification is sent to a given person who must take a given action, for example, **approve** the purchase selection, reject the purchase selection, or place the selection in an "on hold" status. When an **approver** elects to review the **approval** queue, a detailed view of the item to be reviewed may be provided to the **approver**. If a set of items are listed in the **approval** queue, the **approver** may perform a line item veto or **approval** on particular items. It is noted that depending upon the rules in place, one or more **approvers** may be notified to review the request before **approval**.

It is further noted that a given purchase request may be separated into multiple requests due to policy rules, so that automatically approved requests are immediately processed, while others are delayed pending **approval**, if manual **approval** is required. In addition, items within each request may be logically split and managed within the server according to vendor or any other desired feature... ..helps describe example processes that can be part of the order management activities 806. In this embodiment, there are a Requester (Finance Administrative Assistant), a **purchase order** number (PO#) assigner (Widget Purchaser), Purchaser (Widget Purchaser), a Receiver (Finance Administrative Assistant), and a Supplier (Acme Supply Co.). As noted at the bottom of... ..Assigner receives an approved request through node 840, which represents entry from the request management activities 804. This approved request is assumed to have no **purchase order**: associated with it at this point in the process. In block 844, the PO# Assigner adds a PO# to the purchase request. It is noted... ..the PO# can also be provided in other manners, for example, the PO# can be automatically provided by the PM system 100, for example, where **purchase order** numbers (PO#s) are automatically assigned based upon predetermined company rules for PO numbering. In block 846, the Purchaser receives the approved request that now has an associated PO#. In block 848, the Purchaser can review the **purchase order** (PO) details and place the order with the Supplier. In block 856, the Supplier receives the PO through an appropriate medium, such as by fax... ..are made back to the Requester through node 870. If the answer is "YES," then flow proceeds to block 880 where the Accountant closes the **purchase order** (PO) in the accounting system. In block 882, invoice data can be exported to corporate accounting systems, such as company system(s) represented by item... ..information concerning these approved purchase requests, including accounting codes,

automatically into accounting software, reducing entry time and errors.

The purchasing management system 100 thereby streamlines **approval** cycles, so involvement of purchasing and financial managers is reduced to handling policy exceptions, such as requests over a certain dollar amount.

The purchasing and...Example below, dynamic payment cards may be set-up with some level of approved spending without requiring the user to go through the purchase request **approval** process. If a user makes such a transaction, that purchase will typically not have a purchase request or **purchase order** associated with it. Using data provided from the card processing system 130, the purchasing management system 100 can synthesize a purchase request and/or a **purchase order** for this transaction. This synthesized purchase request/order can then be used by a company's accounting system so that the transaction can be reconciled... ..dynamic purchasing rules. The purchasing management system 100, which delivers the ability to dynamically manage the dynamic payment card to allow execution of any company **approved** transaction, controls the **dynamic** payment card constraint and **approval** 46 parameters stored at the card processor or card processing system 130' The dynamic payment card of the present invention also preserves existing purchasing card... ..scenarios where cards with zero available credit could be distributed to a broad set of employees. Once an employee has submitted a request and received **approval**, the purchasing management system 100 can dynamically manage the dynamic payment card through a card processing system 130. When the purchase is initiated, the card processor approves the purchase if it falls within the **approved** parameters. Advantageously, this **dynamic** payment card platform of the present invention resolves control contradictions by enabling a company to choose a high degree of control without sacrificing desired efficiencies... ..card. As these purchases are made, they count against the available pre-approved credit or velocity that has been set up for this card. Upon **approval** of the expenditure or at some other point in the purchasing management process, the purchases can be cleared from the card, thereby increasing or restoring the available credit and purchasing authority for the user's dynamic payment card. The result is a declining **approval** balance type feature and associated purchasing control that provides employees a limited ability to incur company expenditures without personal reimbursement worries, while maintaining dynamically managed... ..can be ultimately processed by the system, while still freeing the employee from having to worry about being reimbursed for ME expenditures.

With this declining **approval** balance feature, therefore, the user can be "trusted" by the company to obligate the company for a finite amount of money. The selection of that... ..below.

In addition, as stated above, post-transaction requests may also be synthesized for expenditures, such as T&E or other expenditures made without pre-**approval** using pre-approved spending authority. For example, with respect to T&E MCC groups where a user's "trusted" velocity is maintained by the purchasing... ..information, may be requested or required from the user with respect to the purchases on the synthesized request, which may then be forwarded on for **approval** or other processing through the purchasing management system. When each justified expense is approved or at another ...and flexible control feature for the purchasing management system 100.



FIG. 2 is a flow diagram for purchase request processing 200 that more generally utilizes **dynamic** payment identifiers and **approval** parameters, according to the present invention. Initially, in block 202, users or -other elements (people, equipment, vehicles, buildings, or any other desired person, place or... ..rejected, flow passes to block 230. As block 230 indicates, a new or modified purchase request is required for a user to continue with obtaining **approval** to complete the desired transaction.

If the purchase request is approved, either in whole or in part, flow passes to block 208, where a **purchase order** is generated. This **purchase order** represents the details of the requested transaction. As a result of the purchase request processing, and as represented in block 208, **approval** parameters are also generated and associated with the purchase request and **purchase order**. These **approval** parameters may include any of a variety of details, including the time within which the, transaction must be completed, approved vendors, approved transaction amounts, or any other desired transaction limitation or requirement. Thus, once approved, each purchase request has associated with it a set of **dynamic approval** parameters, a **purchase order**, and a dynamic payment identifier. This information is provided to block 220 for further processing and reconciliation.

In block 210, the **dynamic approval** parameters are provided to the dynamic payment processing system that will process the transaction, along with the associated dynamic payment identifier and any other desired information. These **approval** parameters, which represent **approval** requirements for a given **purchase order**, are then dynamically stored with respect to the dynamic payment identifier used for the purchase request in block 204. The storing of these **dynamic approval** parameter can be implemented, for example, through a review by the purchasing management system of the **approval** parameters that are associated with the dynamic payment identifier and that are currently stored by the processing system and then through appropriate parameter management commands, such as add, delete or modify commands, sent by the purchasing management system to the processing system to cause to be stored **approval** parameters in a state that will allow the approved transaction to proceed. In block 212, the user initiates a transaction utilizing the dynamic payment identifier, which may be, for example, a dynamic payment card. In block 214, the transaction details are correlated to the **dynamic approval** parameters stored for that **dynamic** payment identifier. It is noted that each dynamic payment identifier may have a plurality of different sets of **approval** parameters with one set being associated with each approved transaction and related **purchase order**. Thus, the correlation that occurs in block 214 would be to identify which set of 49 **approval** parameters should be used for the initiated transaction. For example, vendor information may first be utilized to limit the initiated transaction to a reduced number of the sets of **approval** requirements. From there, transaction amount or product/service types may be utilized to further determine which set of **approval** requirements to utilize. Alternatively, another identifying number or other identifier could be stored and utilized to directly relate initiated transactions with approved **purchase orders**.

In block 216, the transaction is reviewed or processed by the dynamic payment processing system to determine if the transaction falls within the **approval** parameters. If the transaction falls outside of these parameters, flow passes to block 224 where the transaction is rejected. From there, decision block 226 provides.... ..pass back to block 230 wherein a new purchase

request would be required.

Looking back to block 216, if the transaction does fall within the **approval** parameters, flow passes to block 218 where the transaction is completed. The transaction details are provided back to the purchasing management system, and in block 220, the transaction is reconciled with the **purchase order** and other information provided previously from block 208. Once transactions are reconciled, accounting details may be reported to clients, as indicated by block 222. It...a purchase request is received for a product and service. Moving on to block 404, the purchase request is processed and a positive or negative **approval** response is completed. As discussed herein, it is contemplated that the purchase request in block 402 may take a wide variety of forms, from a... price from a particular vendor to a general request for an amount of money to be used to meet some specified need. In turn, the **approval** processing in block 404 considers these widely varying requests and provides a response that may also vary widely in specificity. For example, the **approval** may range from **approval** for a specific 50 product or service from a specific vendor for a specific price to **approval** for an allocation of an amount of money for the purchase of a product or service to meet an approved need. In block 406, the... exercises the accounting role of matching and correlating the various aspects of the purchase, for example, the received products and services, the purchase request, the **approval** conditions, the **purchase order**, and the invoice from the vendor. From this point, in block 426, any necessary payment may be finalized or made to the vendor. It is... range of transactions where purchasing management is desired. As discussed above, the dynamic payment card represents a unique payment identifier that can be associated with **dynamic approval** parameters, including a set of **dynamic approval** parameters for each **approved** purchase request. These **approval** parameters may be, for example, approved vendors, dollar amounts, product types, numbers of items, date or time by which the purchase must be made, or any other desired **approval** criteria. Thus, when a transaction is initiated by an individual possessing a **dynamic** payment card, **approval** of the transaction may be controlled based upon the **approval** information.

associated with the card and the particular purchase request for which the transaction is being initiated. In this way, a company or entity may... requested in the purchase request and regardless of the sales channel from which product or service is to be purchased.

51 In block 410, the **approval** parameters are injected into the card processing system and associated with the appropriate dynamic payment card. In block 412, the user of the dynamic payment... the vendor charges the dynamic payment card. If the charge is approved after correlation and review of the transaction details with the appropriate set of **dynamic approval** parameters for that **dynamic** payment card, flow proceeds along two somewhat parallel paths to blocks 432 and 430. If the transaction is not approved based upon the **approval** information associated with the **dynamic** payment card for that purchase, the process terminates in block 417 based upon the indication of no **approval**.

If the process continues, in block 428, a user may exercise the role of receiver of the products and services, acting to verify receipt of purchase request, the **approval** conditions, the **purchase order**, and the invoice from the vendor. From this point, any necessary payment may also be finalized or made to the vendor. It is again noted... Mary Marketing -- Mary is a member of the marketing department. Mary does not have predictable spending needs. All of Mary's purchases

must have prior **approval** by ACME's VP of Marketing.

Profile for user Ann Admin -- Ami is an administrative assistant who regularly arranges for catered lunch meetings and has... ..Tom at the end of every month. Any individual \$5 purchases over \$5,000 or aggregate monthly spending in excess of \$20,000 require prior **approval** by the CFO. Non-technology purchases require prior **approval** by Tom's supervisor.

Profile for user Sam Shopfloor -- Sam is a facilities engineer for the shop floor. Sam generally does not get involved in purchasing, but has the **approval** of the CFO to make emergency purchases in the case of critical downtime of manufacturing equipment. The CFO has authorized Sam to make emergency expenditures... ..executive who incurs travel and entertainment expenses on a regular basis. Periodically Sally will make purchase requests outside of the T&E category that require **approval**.

#### 5 TABLE 2 - ACME Rocket Products User Profiles

User Name User Type Policy Controls Baseline Card Setup

Mary Marketing Requester No spending without pre- **approval** Credit limit of \$5,000

Card Level: All SICs with aggregate

velocity of 0 transactions and \$0

Ann Admin Requester \$300/month on food service... ..Credit limit \$5,000

service items do not require

justification after the fact; All other Slot 1: Food service SICs with a

spending requires pre- **approval**, velocity of \$300/month

Slot 2: All SICs with aggregate

velocity of 0 transactions and \$0

Ton@ Buyer/ Discretionary spending of Credit limit \$30,000... ..transaction size of \$5,000;

velocity of \$20,000/month and a

Purchases outside of these limits maximum transaction size of \$5,000

require pre- **approval**; All purchases

require both justification and Slot 2: All SICs with aggregate

reconciliation before processing by velocity of 0 transactions and \$0.

accounting.

- 56

User Name User Type Policy Controls Baseline Card Setup

Sam Shopfloor Requester/ All non-emergency spending 2 Cards.

Buyer requires pre-**approval**; Emergency

credit limit of \$10,000 constrained Card 1 - Credit limit \$5@000

to industrial products.

Card Level: All SICs with aggregate

velocity of 0... ..of Credit limit \$10,000

(T&E) \$5,000/month on ME items; All

other purchases require pre- Slot 1: ME category SIC codes with

**approval.** aggregate velocity constraint of \$5000  
Slot 2: All SICs and aggregate velocity of 0 transactions and \$0.

To configure these initial settings at the card...payment, remote order and in-person payment, or in any other desired manner. It is also noted that relating this transaction to FIG. 513, the **approval** process flow would be associated with the reverse flow path 556 because the user or customer has proceeded without first generating a purchase request. **Approval** processing would be expected to occur after the transaction, for example, through purchase request synthesis.

The merchant receives the order in block 604 and communicates in block 606 with a card processor. In decision block 608, the card processor looks to the card settings to make a decision concerning **approval**. In the example shown, the card processor looks to credit and velocity settings for the SIC associated with the initiated purchase. If the transaction does... ..is then notified through the PM system 100 that the transaction has occurred.

The PM system 100 can also synthesize a purchase request and **purchase order** for the transaction in block 618. Because the initiated transaction took advantage of card settings that set up Ann as a buyer role for certain services so that the card had pre-approved spending capabilities, there was no purchase request that was approved prior to the transaction. For accounting and **approval** purposes, it is advantageous for a **purchase order** and request to be generated for this transaction. As such, the PM system 100 synthesizes a purchase request and order from the transaction data... ..not have enough of her monthly food budget remaining to cover this purchase. Ann creates a Purchase Request in the purchasing management system 100 requesting **approval** with the following justification: "Fran, it's been a busy month with all of these customer visits and I have nearly exhausted my catering budget. I need your **approval** to spend an additional \$100 on lunch for the board. Thanks, Ann."

Fran approves this request and, upon **approval**, the purchasing management system 100 checks the available velocity on slot 1 of Ann's dynamic payment card finding that she has \$37.89 of... ..management and control packets can be used to obtain current velocity @ettings and to update those settings.

Ann is informed that her request has been **approved** and her **dynamic** payment card has been enabled for an extra \$100. Ann makes the order with the caterer. Lunch is delivered and the caterer charges Ann's...simplicity. As additional processing, there are blocks that allow for a purchase request through the PM system 100 to be the mechanism for gaining card **approval**. It is noted, therefore, that relating this transaction to FIG. 5B, the **approval** process flow would be associated with either the forward flow path 554 or the reverse flow path 556 depending upon how the user used the card. If a pre-transaction purchase request were utilized, **approval** would be expected to occur prior to the transaction. If pre-approved spending authority were utilized, as with FIG. 6A, **approval** processing would be expected to occur after the transaction, for example, through purchase request synthesis.

In particular, in block 652, the customer fills out a... ..by the PM system 100 according to the

appropriate purchase policies as set up by the company. In block 658, a decision is made to **approve** or not **approve** the request. If the request is not approved, the PM system 100 communicates back to the customer in block 656 that the request has been declined by the **approval** process. If the request is approved, the PM system 100 automatically determines what updates to the card settings will allow the transaction to be approved... ..and that the cost including shipping will be \$1,500.

Mary creates a Purchase Request in the purchasing management system 100 and submits it for **approval**. This request is validated against the company's purchasing policies and it is determined that she needs **approval** from the VP of Marketing. The VP of Marketing is immediately notified, for example through e-mail or phone call, that he has a purchase request to **approve** and approves this purchase.

Upon **approval** of this purchase request, the purchasing management system 100 increases the aggregate card level velocity of Mary's dynamic payment card by one transaction and... ..management and control packets can be used to obtain current velocity settings and to update those settings.

Mary is informed that her request has been **approved** and that her **dynamic** payment card has been enabled for a \$1,500 purchase. Mary places the order with the supplier and uses her dynamic payment card for payment...with his dynamic payment card number from prior purchases, and configures the order, which totals \$28,288. Tom knows that his maximum purchase without pre-**approval** is \$5,000. So, Tom creates an Urgent Purchase Request in the purchasing management system 100, marks it Urgent and includes the following justification: "Fran, Our Dell server was destroyed by a ruptured water line located... ..above our data center. I've configured a replacement on the Dell web site and they can have it here in 24 hours. Need your **approval** to spend the \$28,288. Tom."

63 Because Tom's request is marked Urgent, the purchasing management system 100 can send a text page and... ..new server it will be impossible to verify that the guidance systems are working on the new rocket. Fran subsequently approves this request and upon **approval** the purchasing management system 100 increases the slot 1 velocity and maximum transaction amount on Tom's dynamic payment card to cover the \$28,288... ..management and control packets can be used to obtain current velocity settings and to update those settings.

Tom is informed that his request has been **approved** and his **dynamic** payment card has been enabled for a \$28,288 purchase. Tom returns to the Dell site and completes the checkout process.

Dell builds the server...the total price including tax and shipping will be \$498 Sam creates a Purchase Request in the purchasing management system 100 and submits it for **approval**.

Fran has set an **approval** policy that will auto **approve** purchases under \$500 if there is departmental budget remaining in the appropriate cost center. The purchasing management system 100 determines that there is sufficient headroom in the facilities budget and it is under \$500 and autoapproves this purchase. Upon **approval** of this purchase request, the purchasing management system 100 increases the aggregate velocity on slot 1 of Sam's primary dynamic payment card by one... ..management and control packets can be used to obtain current velocity

settings and to update those settings.

Sam is informed that his request has been **approved** and his **dynamic** payment card has been enabled for a \$498.76 purchase. Sam goes back to Grainger.com and finishes checking out with his dynamic payment card...the associated headroom reinstated on her dynamic payment card. Sally identifies all of the charges for her Canadian visit, justifies them appropriately and submits for **approval** via the purchasing management system 100. Upon **approval**, the purchasing management system 100 reinstates the aggregate velocity available on slot 1 of Sally's dynamic payment card by the appropriate amount.

Again, to... ..processor determines if the initiated transaction falls within card parameters, such as credit and velocity settings. If the answer is in block 704 "no," the **approval** is declined in block 706. If the answer in block 704 is "yes," then process flows continues to block 708 and on to FIG. 7B... ..nines if the initiated transaction falls within card parameters, such as credit and velocity settings. If the answer in block 608 is "no," then the **approval** is declined in block 610. If the answer in block 608 is "yes," then the processing continues on to block 612. It is noted that... ..sales team for the sales kickoff meeting will cost \$375. Sally creates an Purchase Request in the purchasing management system 100 and submits it for **approval**. This request is validated against the company's purchasing policies and it is determined that she needs **approval** from the VP of Sales. The VP of Sales is immediately notified that she has a purchase request to **approve** and approves this purchase. Upon **approval** of this purchase request, the purchasing management system 100 increases the aggregate velocity on slot 2 of Sally's dynamic payment card by one transaction... ..management and control packets can be used to obtain current velocity settings and to update those settings.

Sally is informed that her request has been **approved** and her **dynamic** payment card

### Claims:

...control settings, comprising:

receiving a purchase request from a requestor within an entity;processing the purchase request with respect to purchase policies for the entity;**approving** the purchase request if the purchase policies are satisfied;reviewing control settings for a payment card associated with the approved purchase request;andadjusting the... ..one or more server systems.

3 The method of claim 2, wherein the customizable purchasing management rules comprise an ability to configure organization structures and **approval** chains.

4 The method of claim 2, further comprising notifying an **approver** of a purchase request, if some action is required from the **approver** for the purchase request to be approved, and allowing the **approver** to take the required action through a network accessible **approval** mechanism.

5 The method of claim 2, wherein the network comprises the Internet.

6 The method of claim 1, further comprising associating a payment card... ..of -claim 1, wherein a card processor stores the payment card control settings. - 71 . The method of claim 8, further

comprising generating a set of **approval** parameters for the approved purchase request and comparing the **approval** parameters with the control settings to determine what adjustments to make so that the purchase may be made with the payment card.

10 The method of claim 9, wherein the control settings correspond to the **approval** parameters.

11 The method of claim 8, further comprising comparing parameters of an attempted purchase transaction with the control settings and authorizing the purchase transaction... ..certain purchasing capabilities with respect to a payment mechanism; communicating from the one or more server systems to a processing system to obtain information representing **approval** parameters associated with the payment mechanism, the processing system being a system that processes transactions initiated using the payment mechanism based upon **approval** parameters associated with the payment mechanism, the **approval** parameters being stored by the processing system; and sending from the one or more server systems to the processing system adjustment instructions to adjust the **approval** parameters for the payment mechanism so that the purchasing capabilities are available. 15 18. The method of claim 17, wherein the request comprises an... ..generated by receiving a purchase request from a requestor within the entity, processing the purchase request with respect to purchase policies for the entity, and **approving** the purchase request if the purchase policies are satisfied; and generating the request based upon the purchase request.

19 The method of claim 18, further... ..20 The method of claim 17, wherein the request comprises a request to provide pre-approved purchasing authority for the payment mechanism and wherein the **approval** parameters are adjusted to provide this pre-approved purchasing authority.

21 The method of claim 20, further comprising receiving at the one or more server... ..the transaction data.

22 The method of claim 21, further comprising processing the synthesized purchase request with respect to purchase policies for the entity and **approving** the synthesized purchase request if the purchase policies are satisfied. 73 . The method of claim 20, further comprising adjusting the **approval** parameters to restore the pre-approved purchasing authority based upon an occurrence of one or more selected events after completion of the transaction.

24 The...The method of claim 17, wherein the payment mechanism comprises a payment card, wherein the processing system comprises a payment card processing system, wherein the **approval** parameters comprise control settings for the payment card, and wherein the one or more server systems comprise a purchasing management system.

26 The method of... ..ft.

37 The method of claim 36, wherein the request comprises a purchase request that has been approved based upon entity purchasing policies, wherein the **approval** parameters comprise a set of **dynamic approval** parameters associated with the approved purchase request, and wherein the processing system stores the set of **dynamic approval** parameters for the **approved** purchase request.

38 The method of claim 37, further comprising receiving a plurality of purchase requests associated with a particular dynamic payment identifier, and further comprising communicating a set of **dynamic approval** parameters for each purchase request from the one or more server systems to the processing system, each set of **dynamic approval** parameters being stored by the processing system.

39 The method of claim 38, further comprising comparing parameters of an attempted purchase transaction made using the dynamic payment identifier with the sets of **dynamic approval** parameters stored for that **dynamic** payment identifier, and authorizing the purchase transaction if the parameters match at least one set of **dynamic approval** parameters, the comparing and authorizing steps being conducted by the card processing system. 75 . A system for dynamically managing payment card control settings, comprising one... ..to receive a purchase request from a requestor within an entity, to process the purchase request with respect to purchase policies for the entity, to **approve** the purchase request if the purchase policies are satisfied, to review control settings for a payment card associated with the approved purchase request, and to... ..one or more server systems.

42 The system of claim 41, wherein the customizable purchasing management rules comprise an ability to configure organization structures and **approval** chains. 1 S 43. The system of claim 41, wherein the one or more server systems are further configured to notify an **approver** of a purchase request, if some action is required from the **approver** for the purchase request to be approved, and further comprising a network accessible **approval** mechanism through which the **approver** can take the required action.

44 The system of claim 43, wherein the network comprises the Internet.

45 The system of claim 40, wherein the... ..payment card control settings.

48 The system of claim 47, wherein the one or more server systems are further configured to generate a set of **approval** parameters for the approved purchase request and to compare the **approval** parameters with the control settings to determine what adjustments to make so that the purchase may be made with the payment card. 76 . The system of claim 48, wherein the control settings correspond to the **approval** parameters. so. The system of claim 47, wherein the card processor is configured to compare parameters of an attempted purchase transaction with the control settings...within an entity to make available certain purchasing capabilities with respect to a payment mechanism; to communicate with a processing system to obtain information representing **approval** parameters associated with the 77 payment mechanism that are stored and used by the processing system to process transactions initiated using the payment mechanism based upon the **approval** parameters associated with the payment mechanism; and to send to the processing system adjustment instructions to adjust the **approval** parameters for the payment mechanism so that the purchasing capabilities are available.

57 The server-based system of claim 56, wherein the request comprises an... ..to receive a purchase request from a requestor within the entity, to process the purchase request with respect to purchase policies for the entity, to **approve** the purchase request if the purchase policies are



satisfied, and to generate the request based upon the purchase request.

58 The server-based system of... ...The server-based system of claim 56, wherein the request comprises a request to provide preapproved purchasing authority for the payment mechanism and wherein the **approval** parameters are adjusted to provide this pre-approved purchasing authority.

60 The server-based system of claim 59, wherein the one or more server systems... ...the one or more server systems is further configured to process the synthesized purchase request with respect to purchase policies for the entity and to **approve** the synthesized purchase request if the purchase policies are satisfied.

62 The server-based system of claim 59, wherein the one or more server systems is further configured to send adjustment instructions to adjust the **approval** parameters to restore the preapproved purchasing authority based upon an occurrence of one or more selected events after completion of the transaction.

63 The server... ...server-based system of claim 56, wherein the payment mechanism comprises payment cards, wherein the processing system comprises a payment card processing system, wherein the **approval** parameters comprise control settings for the payment card, and wherein the one or more server systems comprise a purchasing management system.

65 The server-based...The server-based system of claim 75, wherein the request comprises a purchase request that has been approved based upon entity purchasing policies, wherein the **approval** parameters comprise a set of **dynamic approval** parameters associated with the approved purchase request, and wherein the processing system is configured to store the set of **dynamic approval** parameters for the **approved** purchase request.

77 The server-based system of claim 76, wherein the one or more server systems are further configured to receive a plurality of purchase requests associated with a particular dynamic payment identifier and to communicate a set of **dynamic approval** parameters for each purchase request from the one or more server systems to the processing system, each set of **dynamic approval** parameters being stored by the processing system.

78 The server-based system of claim 77, wherein the card processing system is further configured to compare parameters of an attempted purchase transaction made using the dynamic payment identifier with the sets of **dynamic approval** parameters stored for that **dynamic** payment identifier and to authorize the purchase transaction if the parameters match at least one set of **dynamic approval** parameters. 80

00831835

**AUTOMATED METHOD AND SYSTEM FOR SELECTING AND PROCURING  
ELECTRONIC COMPONENTS USED IN CIRCUIT AND CHIP DESIGNS  
PROCEDE ET SYSTEME AUTOMATISES POUR SELECTIONNER ET OBTENIR DES  
COMPOSANTS ELECTRONIQUES UTILISES DANS LA CONCEPTION DE CIRCUITS  
INTEGRES ET DE PUCES**

**Patent Applicant/Patent Assignee:**

- **CADENCE DESIGN SYSTEMS INC**  
2655 Seely Avenue, San Jose, CA 95134; US; US(Residence); US(Nationality)

**Inventor(s):**

- **ROBERTSON William H**  
1574 Kathy Lane, Los Altos, CA 94024; US
- **PLYMALE James M**  
10145 SW Redwing Terrace, Beaverton, OR 97007; US

**Legal Representative:**

- **VANDERLAAN Christopher A(et al)(agent)**  
Lyon & Lyon LLP, Suite 4700, 633 West Fifth Street, Los Angeles, CA 90071; US

	Country	Number	Kind	Date
Patent	WO	200165423	A2-A3	<b>20010907</b>
Application	WO	2001US6155		20010226
Priorities	US	2000514674		20000228

**Designated States:** (Protection type is "Patent" unless otherwise stated - for applications prior to 2004)

AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG,  
BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE,  
DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH,  
GM, HR, HU, ID, IL, IN, IS, JP, KE, KG,  
KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV,  
MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ,  
PL, PT, RO, RU, SD, SE, SG, SI, SK, SL,  
TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU,  
ZA, ZW

[EP] AT; BE; CH; CY; DE; DK; ES; FI; FR; GB;  
GR; IE; IT; LU; MC; NL; PT; SE; TR;

[OA] BF; BJ; CF; CG; CI; CM; GA; GN; GW; ML;  
MR; NE; SN; TD; TG;

[AP] GH; GM; KE; LS; MW; MZ; SD; SL; SZ; TZ;  
UG; ZW;

[EA] AM; AZ; BY; KG; KZ; MD; RU; TJ; TM;

**Language** Publication Language: English

Filing Language: English

Fulltext word count: 8239

### **English Abstract:**

...or a remote supplier or manufacturer database. As dynamic parts are selected for use in a design, their information may be forwarded to a parts **approval** process for qualification with the designer's organization. From a design in the schematic program having multiple dynamic parts, a bill of materials may be...

### **Detailed Description:**

...easily result in additional design time and rework. If the designer is allowed to use a new part@ it must first be approved, but the **approval** process can be cumbersome, bureaucratic and lengthy, adding to the overall time to market.

Other aspects of managing electronic component data include documentation and configuration... engineering design. These N4RP systems typically contain a great number of parts identified by part number, and are utilized with the intent of speeding up **approval**, purchasing and procurement of parts. However, because NW systems generally do not include the type of detailed specification data that an engineer requires to perform... ..to maintain a part in an inventory system can be quite costly as well (e.g., on the order of \$10,000 per year). Further, **approval** and qualification of new parts and new suppliers is time-consuming and tedious. Thus, the failure for an engineer to use preapproved parts in the...embodiment, when a dynamic part is placed into the user's schematic program, the dynamic part is also automatically entered into the user's parts **approval** system.

In yet another embodiment, an electronic bill of materials containing a list of the dynamic parts utilized in the design and their quantities is... ..may be linked to a local and/or remote database in order to provide current component information regarding these parts (such as, for example, their **approval** status and backlog time).

Further embodiments, variations, modifications and enhancements are also described herein and illustrated in the drawings.

### **BRIEF DESCRIPTION OF DRAWINGS**

The invention...remote database client, with the remote database 304 acting as a remote server.

In a preferred embodiment, further steps are carried out to enter selected **dynamic** parts in a parts **approval** process and/or procurement process. Accordingly, in a next step 614 of the design and procurement process 600 illustrated in FIG. 5, when a dynamic part 460 is placed in the local database 415 or 430, the dynamic part 460 enters a parts **approval** process, so as to qualify it for use by the designer's organization. Preferably, the parts **approval** process is handled through the resource planning database 410. However, the parts **approval** process may be carried out in any of a variety of different manners depending on the nature of the user's organization, and may comprise, for example, an automated process, a semiautomated process, or a manual process. By automatically placing a dynamic part 460 into the parts **approval** process as soon as the dynamic part 460 is selected for use in the schematic 15 program 404, the parts **approval** process begins as promptly as possible, reducing delays in the design development process. Further, problems associated with a designer's failure to identify, for **approval** purposes, all of the electronic components used in the design are minimized, because all dynamic parts 460 are automatically entered into the parts **approval** process upon placement in the schematic program 404.

In a next step 616, after the dynamic part is placed in the parts **approval** process, the process checks for the existence of the dynamic part 460 within the resource planning database 410. If the **dynamic** part 460 is already **approved** for use in design and manufacturing, then the parts **approval** process is complete, and the process moves to step 618. If, on the other hand, the dynamic part 460 is not already approved (i.e., ...part is found in the resource planning database 410, then the process moves to step 624 in which the designer is notified that the original **dynamic** part 460 is not **approved**, but that an equivalent pre-approved part exists. The designer may then be encouraged, or required, to use the equivalent pre-approved part. If there is no equivalent part in the resource planning database 410 in step 620, then the process moves to step 622, wherein the parts **approval** process continues. From this point forward, the parts **approval** process may include many other steps which will vary from company to company, and which may include such items as quality certification, site visits to bill of materials also includes information regarding the **approval** status of these parts and the current lead time and cost for each of the parts. The lead time and cost may be obtained directly.... (237) filed concurrently herewith, previously incorporated by reference as if set forth fully herein. The procurement process may involve, for example, the automatic generation of **purchase orders** for the desired parts in the bill of materials (based on the expected quantity of production), which may be transmitted in electronic form over the...

## Claims:

...of claim 7, wherein said local database comprises a resource planning database, said method further comprising the steps of entering a component represented by said **dynamic** part into a parts **approval** process, and comparing the component with data records of components already stored in said resource planning database.

9 The method of claim 7, further comprising....system of claim 19, wherein said local database comprises a resource planning database containing data records of approved parts, said system

flirther comprising a parts **approval** process for comparing each **dynamic** part transmitted to the user computer with said data records of approved parts.

21 The system of claim 18, wherein one or more of said...

8/3K/6 (Item 3 from file: 349)  
DIALOG(R)File 349: PCT FULLTEXT  
(c) 2011 WIPO/Thomson. All rights reserved.

00783601

**METHOD AND SYSTEM FOR PROCESS INTERACTION AMONG A GROUP**  
**PROCEDE ET SYSTEME PERMETTANT UN PROCESSUS D'INTERACTION DANS UN**  
**GROUPE**

**Patent Applicant/Patent Assignee:**

- **FIREDROP INC**  
Suite 201, 3000 Bridge Parkway, Redwood City, CA 94065; US; US(Residence);  
US(Nationality); (For all designated states except: US)

**Patent Applicant/Inventor:**

- **MILLER Graham**  
San Francisco, CA; US; US(Residence); US(Nationality); (Designated only for: US)
- **HANSON Michael**  
Menlo Park, CA; US; US(Residence); US(Nationality); (Designated only for: US)
- **AXE Brian**  
San Francisco, CA; US; US(Residence); US(Nationality); (Designated only for: US)
- **EVANS Steven Richard**  
Los Altos Hills, CA; US; US(Residence); US(Nationality); (Designated only for: US)

**Legal Representative:**

- **VYAS Shekhar (agent)**  
Fish & Richardson P.C., Suite 500, 4350 La Jolla Village Drive, San Diego, CA 92122;  
US

	Country	Number	Kind	Date
Patent	WO	200117172	A1	20010308
Application	WO	2000US23221		20000823

	Country	Number	Kind	Date
Priorities	US	99151476		19990830
	US	99151650		19990831
	US	99427378		19991025
	US	99426648		19991025
	US	99427152		19991025
	US	2000483508		20000114

**Designated States:** (Protection type is "Patent" unless otherwise stated - for applications prior to 2004)

AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG,  
BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE,  
DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH,  
GM, HR, HU, ID, IL, IN, IS, JP, KE, KG,  
KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV,  
MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ,  
PL, PT, RO, RU, SD, SE, SG, SI, SK, SL,  
TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN,  
YU, ZA, ZW

[EP] AT; BE; CH; CY; DE; DK; ES; FI; FR; GB;  
GR; IE; IT; LU; MC; NL; PT; SE;

[OA] BF; BJ; CF; CG; CI; CM; GA; GN; GW; ML;  
MR; NE; SN; TD; TG;

[AP] GH; GM; KE; LS; MW; MZ; SD; SL; SZ; TZ;  
UG; ZW;

[EA] AM; AZ; BY; KG; KZ; MD; RU; TJ; TM;

**Language** Publication Language: English

Filing Language: English

Fulltext word count: 15056

### **Detailed Description:**

...in the database.

Implementations of the invention may include one or more of the following. The process interaction may be a discussion, tracking, purchase, collection, **approval**, or a negotiation. The **dynamic** content may include concatenated text, links, buttons, and 4 graphics. An external source in data communication with the server may be used to deliver content... ...16 illustrates an example of an updated image of the zaplet of FIG. 15.

FIG. 17 is an example of an electronic form for an **approval** process.

FIG. 18 illustrates an example image of a zaplet for supporting an **approval** process among a group of participants.

FIG. 19 illustrates an example of an updated image of the zaplet of FIG. 18.

FIG. 20 is an...systems and methods may be used to enable participants to interact within a specific process. These processes may include a - 24 discussion, tracking, purchase, collection, **approvals**, or negotiations. A discussion can be defined as the exploration of some topic, series of topics, or sub topics. Tracking can be defined as a... ..a decision process proceeds before a completed purchase. A collection may be defined as the collection of money or other commitments from other participants. An **approval** may be defined as a collection of **approvals** from possible multiple participants. The order for the collection of **approvals** may be predetermined. A negotiation may be defined as an auction, feedback collection and issue resolution, or a "haggling" tool.

I 0 An example of...the participant purchasing the tickets can determine, for example, who is attending the event.

1 5 The preferred zaplet may also be used as an **approval** process tool. The zaplet can be used to generate a process of **approval** in which multiple individuals within an enterprise environment must **approve** or disallow, for example, a **purchase order** requisition submitted by a potential purchaser.

Initialiv. one of the participants accesses a live electronic form 1200 (FIG. 17) by executing the step 501. The...to the 601, 602, 603). and 605 described above. The electronic forin 1200 may also include an identification field 1204 to receive, for example, a **purchase order** number.

A region 1206 may be used to describe information relevant to the **purchase order**.

For example, the region 1206 may include the invoice amount, the payment due date, the **approval** due date, the purchase description, the quantity, the list price, and the potential buyer and seller information. The form 1200 may be completed by an initiating participant or automatically generated by a company, %,"spurchasin software that is in data  
9  
communication with the server 6.

The electronic forin 1200 also includes an **approval** region 1207. This region lists all participants or individuals within a company who must **approve** the **purchase order**. Once the initiating participant has completed the electronic form 1200, the form 1200 is submitted to I 0 the server 6 in the step 502... ..The information region 1301 may include information from the region 1206, the fields 1201, 1202, 1204, and 1205. In this way, a participant who must **approve** the **purchase order** can view specific details about the purchase.

The zaplet 1300 may also include an interaction region 1315 similar to the interaction region 915 described above... The zaplet 1300 may also include a dynamic content region 1310. The dynamic content region may include a list of participants required to **approve the purchase order** and buttons that can be selected to **approve** or disapprove the **purchase order**.

FIG. 19 shows an updated image of the zaplet 1300. As shown in FIG. 19, the dynamic content region 110 is updated with **approvals** or disapprovals from the individuals who must be notified of the **purchase order** before **approval** is given to the initiating participant. In a preferred configuration, the **approval** zaplet is distributed and follows a defined sequential or hierarchical path. This means that a participant who must **approve** the transaction at one level, for example, an initiating participant's direct manager, must first **approve** the **purchase order** request before the zaplet 1300 is passed to the next **approval** level, such as a participant in the accounting department. The zaplet, in turn, may then be passed to the next higher **approval** level. In some configurations, a participant who receives the **purchase order approval** zaplet for **approval** may only receive the zaplet if those participants at the lower levels of the hierarchical path have approved the **purchase order**.

10 Preferably, a participant who receives the zaplet can view the **approval** or disapproval of the **purchase order** using the dynamic content region 1310. In one configuration, if a disapproval is received, the **approval** process may be terminated and the request can be marked disallowed and returned to the initiating participant.

In another example, the zaplet can be used...

#### Claims:

...the information identifying the item to be collected automatically by an external source in data communication with the server. - 50

32 A method for gathering **approvals**, comprising:  
generating an electronic form having a plurality of network addresses associated with participants having an **approval** powersending the electronic form to a server;parsing an electronic message received by at least one of the participants from the server in response... the electronic medium being stored in the server and the dynamic content regions including a then current dynamic content representative of a status of the **approval**; and 10 asynchronously **dynamically** updating and **dynamically** retrieving the **approval** status 1 from the server based on input from any of the participants.

33 The method of claim 32, wherein at least one of the dynamic content regions further comprises an updated **approval** status.

34 The method of claim 32, wherein the electronic medium further comprises information about an item to be approved.

35 The method of claim 2, wherein the serving step further comprises serving the electronic medium to any of the participants when an **approval** is received by a prior participant. - 51



36 A method for conducting an auction, comprising:  
generating an electronic form having a plurality of network addresses...

8/3K/7 (Item 4 from file: 349)  
DIALOG(R)File 349: PCT FULLTEXT  
(c) 2011 WIPO/Thomson. All rights reserved.

00775310

**A SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR DETERMINING  
CAPABILITY LEVELS OF A RELEASE MANAGEMENT PROCESS AREA FOR  
PROCESS ASSESSMENT PURPOSES IN AN OPERATIONAL MATURITY  
INVESTIGATION**

SYSTEME, PROCEDE ET ARTICLE MANUFACTURE POUR DETERMINER LES  
NIVEAUX DE CAPACITE D'UNE ZONE DU PROCESSUS DE GESTION DE DIFFUSION  
A DES FINS D'EVALUATION DE PROCESSUS DANS UNE ETUDE DE MATURITE  
OPERATIONNELLE

**Patent Applicant/Patent Assignee:**

- **ACCENTURE LLP**  
1661 Page Mill Road, Palo Alto, CA 94304; US; US(Residence); US(Nationality); (For  
all designated states except: US)

**Patent Applicant/Inventor:**

- **GREENBERG Nancy S**  
5529 Newton Avenue South, Minneapolis, MN 55410; US; US(Residence);  
US(Nationality); (Designated only for: US)
- **WINN Colleen R**  
11472 Fairfield Road #103, Minnetonka, MN 55305; US; US(Residence);  
US(Nationality); (Designated only for: US)

**Legal Representative:**

- **HICKMAN Paul L (agent)**  
Oppenheimer Wolff & Donnelly LLP, 1400 Page Mill Road, Palo Alto, CA 94304; US

	Country	Number	Kind	Date
Patent	WO	200108074	A2	20010201
Application	WO	2000US20278		20000726

	Country	Number	Kind	Date
Priorities	US	99361335		19990726

**Designated States:** (Protection type is "Patent" unless otherwise stated - for applications prior to 2004)

AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG,  
BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE,  
DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH,  
GM, HR, HU, ID, IL, IN, IS, JP, KE, KG,  
KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV,  
MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ,  
PL, PT, RO, RU, SD, SE, SG, SI, SK, SL,  
TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN,  
YU, ZA, ZW

[EP] AT; BE; CH; CY; DE; DK; ES; FI; FR; GB;  
GR; IE; IT; LU; MC; NL; PT; SE;

[OA] BF; BJ; CF; CG; CI; CM; GA; GN; GW; ML;  
MR; NE; SN; TD; TG;

[AP] GH; GM; KE; LS; MW; MZ; SD; SL; SZ; TZ;  
UG; ZW;

[EA] AM; AZ; BY; KG; KZ; MD; RU; TJ; TM;

**Language** Publication Language: English

Filing Language: English

Fulltext word count: 85690

### Detailed Description:

...industry-recognized language for "programming the Internet." Sun defines Java as: "a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, **dynamic**, buzzwordcompliant, general-purpose programming language. Java supports programming for the Internet in the form of platform-independent Java applets." Java applets are small, specialized applications...Improvements may be based on incremental operational refinements or through innovations, such new technologies. Improvements may typically be driven by the following activities.

Identifying and **improving** changes to the standard process definition on the basis of quantitative understanding of the process.

Providing adequate resources to effectively implement the approved changes in...the Service Desk's assessment of incident and problem management?

Base Practice: 1.11 Receive Requests

Are requests handled immediately or do they require provisioning/**approval**? Does the Service Desk coordinate the **approval** of requests with the appropriate functions and notify requester of **approval** /rejection?

If request requires functions outside the Service Desk, how does the Service Desk pass responsibility to the appropriate personnel?

Do SLAs exist between the...or the affected users for a specified amount of time.

f

BP Number 1.7

BP Name Agree to SLAs with users

BPI) ripfill,

on, Final **approval** should be obtained from the parties involved in the SLA agreement process.

This will be representatives from the IT organization, client management, and the users...document (eg. memo) describing it is available.

1.1.7 Agree to SLAs with users Prior to being finalized, SLAs were reviewed with users and their **approval** obtained. Signed off SLA document exists.

1.1.8 Report on SLA performance Regular reporting on SLA performance occurs.

Reports include predefined metrics and KPIs... budgeting or costing that is used in SLA management?

Base Practice: 1.7 Agree to SLAs with Users

To what parties are SLAs submitted for **approval**?

How is **approval** of the SLA documented?

Where is information about the finalized SLA stored? Are SLA summaries available to users? Is there a system for users to...Are SLAs drafted based on the requirements identified?

Is a charge back structure defined and used to assign rewards/penalties based on SLA performance?

Is **approval** of the draft SLA obtained from all parties involved?

Are KPIs and metrics regularly reported to monitor SLA performance?

Process Area 1.1 SLA Management...the OLA with the vendor cannot equal or exceed 24 hours.

BP Number, 1.6

BP Name Agree to OLAs with suppliers

BP Description Final **approval** should be obtained from the parties involved in the OLA agreement process.

This will include suppliers, information service personnel, and possibly representatives from the user...any provisions for rewards if OLA requirements are exceeded?

Base Practice: 1.6 Agree to OLAs with Suppliers

To what parties are OLAs submitted for **approval**?

How is **approval** of the OLA documented?

What is the approximate proportion of OLAs created with internal service providers compared to external suppliers?

Where is information about the...select service providers from the pool of candidates?

For each service provider, is an OLA prepared and are measures for assessing service levels specified?

Is **approval** of the OLA obtained from all parties involved in the OLA agreement process?

Are service levels measured and reported periodically to monitor OLA compliance?

Are...the Service

Desk's assessment of incident and problem management?

Base Practice: 1.11 Receive Requests

Are requests handled immediately or do they require provisioning/**approval**? Does the Service Desk coordinate the **approval** of requests with the appropriate functions and notify requester of approval/rejection?

If request requires functions outside the Service Desk, how does the Service Desk pass...planned Example of budgets and cost allocation plans that are budgets and cost allocation plans submitted to management and user representatives for with management/user review/**approval**.

1.13 Define what the reporting \*when questioned, the policies on financial information specifications are and report on to be reported can be described.

financial...user representatives for review?

Does management or customer sponsor sign off plans?

What is the procedure for modification of the budgets or plans if initial **approval** is not obtained? Is the cost allocation plan reviewed to ensure that costs are accurately captured and prices are fair?

How does this occur?

Base... Change Control

Plan, Human Resources, Operations Personnel)

2. Are current procedures and resources periodically assessed with the intent to promote continuous improvement? What is the **approval** process for proposed solutions? Are these solutions evaluated for impact?

3. Are there regularly scheduled training programs that address User Administration procedures? If

so, what...ad hoc production orders come through many considerations, such as other processes p M

running (backup/restoration, software distribution, etc.), need to be weighed before **approving** and sending the ad hoc order.

diffy the master

When ad hoc requests are considered procedures/requirements are used to mo

,g ample,,,,,  
computer schedule.

111-P...feedback in Print management receives  
order to maintain feedback via, e-mail, reports and  
knowledge and experience meetings from customers and other  
process areas regarding issues,  
**approvals** and reviews.

#### Level 4 Assessment Indicators

Process Attribute Generic Practice Example of Assessment Indicator Assessment  
Indicators  
at Client

89

Process GP4.1 Establish Print management...Print forms

1. Does the output/print management personnel review forms prior to their use throughout the distributed system? Reports? How is this process of **approval** managed (e.g. meetings, requests, sample stock test runs, etc.)?

2. Can forms be collated and packed?

3. Are there certain printers on the network...established directory management and address servicing procedures?

2. Are current resources and procedures periodically assessed with the intent to promote continuous improvement? What is the **approval** process for proposed solutions? Are all potential stakeholders involve in the decision process? How often are these solutions implemented and by whom? 3. How are...that backup/restore provisions are made primarily to handle disasters rather than for restoring files accidentally deleted or overwritten. However 'special requests for restoring individual files are processed through the Service Desk with prior **approval**.

After analyzing the request, the correct tape is determined and the corresponding directory files are restored If necessary, changes are applied from the journal to the tired copy...Practice: 2 1

#### Define Security Objectives

What types of issues are covered by the formal security policy?

Was the security policy submitted to management for **approval**?

Is the security policy documented and available to customers and management?

Base Practice: 2 2 Develop security plan and policies

Please describe the contents of...Emergency, which require immediate attention and correction/implementation.

#### PAIIs"Bifs'e 3 1 Change Initiation

„Practices 3 2 Change Impact Analysis/Assessment

3 3 Change **Approval**

3 4 Change Communication and Scheduling

3 5 Change Implementation Planning and Preparation

3 6 Change Request Tracking

3 7 Change Implementation

3 8 Change...levels impacted by the proposed change. In

addition, dependencies to other applications, databases or systems are identified

BP Number 3 3

### BP Name **Change Approval**

BP Description Authority to proceed with a Change Request can come from various sources depending on the type of change. The organization must be able... ..request requires the Change Control Coordinator's or the Managing Director's signature before it will be implemented. It is up to the person giving **approval** to clarify any concerns regarding the change request with the requestor (or the business leader, as appropriate). If necessary given the urgency of other requests...are recorded at time of receipt.

3 2 Change Impact Impact analysis template or an example of completed Analysis/Assessment analysis report.

3 3 Change **Approval** Example of completed change request form shows authorization by appropriate personnel.

3 4 Change Communication and A system for coordinating the planned date of a Scheduling... repeatedly

Process GP3.3: Plan for human Change requests are always handled

Resource resources proactively according to the documented policy (eg.

requests are not processed without appropriate **approval**).

GP3.4 Provide feedback in On completion of a change request, order to maintain knowledge feedback is solicited from the requester.

and experience

Level 4 Assessment...consequences of the change impact analysis (i.e. is the change request rejected if the

change analysis yields particular results)?

Base Practice: 3 3 Change **Approval**

Whose **approval** is needed before a change request can be implemented? Does the person(s) whose **approval** is necessary depend on the scope or priority level of the change?

How is **approval** obtained and documented?

Is the change requestor notified of change **approval** or rejection?

Base Practice: 3 4 Change Communication and Scheduling

Once **approval** is obtained, what is the process for estimating the time and scheduling the change? Are other completion times and dates factored into the estimated time ...submitted?

Prior to implementing a proposed change, is an impact analysis performed to identify the effects of the proposed change?

Does a change request require **approval** from designated personnel before being implemented?

Are the implementation time and completion date estimated and scheduled?

Are all affected parties notified of a scheduled change...taking place. If changes to the schedule are required, Deployment is responsible for coordinating the changes across all of the groups involved and seek management **approval** for the changes.

PA's Base Confirm schedule with all key groups periodically  
Practices Determine whether schedule will be impacted based on issues or problems... ..and external groups may result in a scheduling conflict or a need for longer lead time. The deployment schedule should accommodate this issue and managerial **approval** should be gained.

Example If an external vendor cannot complete hardware installations prior to scheduled deployment, the deployment schedule should be changed accordingly and management should **approve** prior to any advancement.

BP Number 3 4

BP Name Report on progress of deployment plan

PP Description The current status of deployment and any...Assessment Indicator Assessment Indicators  
at Client

3 1 Confirm schedule with all Feedback is received during a meeting with payroll key groups periodically noting the **approval** of a May deployment, as the new date is after the tax season. An electronic or hard copy schedule noting prep, training and customer types is... ..5 Maintain Feedback might be collected via meetings  
communication among team and reports from physical planning and members management regarding lead times. GP2.7 Employ version **Approvals** are gained on schedules to control to manage changes to accommodate the latest issues noted on Level 3  
Assessment Indicators  
159

Process Attribute Generic Practice...taking place. If changes to the schedule are required, Deployment is responsible for coordinating the changes across all of the groups involved and seek management **approval** for the changes.

Questionnaire

Process Area 3.4 Deployment

Yes No Don't N/A

Know

I Are deployment schedules confirmed with all key groups...

8/3K/8 (Item 5 from file: 349)

DIALOG(R)File 349: PCT FULLTEXT

(c) 2011 WIPO/Thomson. All rights reserved.

00775307

**A SYSTEM, METHOD AND COMPUTER PROGRAM FOR DETERMINING  
CAPABILITY LEVELS OF PROCESSES TO EVALUATE OPERATIONAL  
MATURITY OF AN ORGANIZATION**  
SYSTEME, PROCEDE ET ARTICLE DE FABRICATION DESTINES A DETERMINER DES  
NIVEAUX DE CAPACITE D'OPERATIONS POUR DES BESOINS D'EVALUATION  
D'OPERATION DANS UNE RECHERCHE DE MATURITE OPERATIONNELLE

**Patent Applicant/Patent Assignee:**

- **ACCENTURE LLP**  
1661 Page Mill Road, Palo Alto, CA 94304; US; US(Residence); US(Nationality); (For all designated states except: US)

**Patent Applicant/Inventor:**

- **GREENBERG Nancy S**  
5529 Newton Avenue South, Minneapolis, MN 55410; US; US(Residence); US(Nationality); (Designated only for: US)
- **WINN Colleen R**  
11472 Fairfield Road #103, Minnetonka, MN 55305; US; US(Residence); US(Nationality); (Designated only for: US)

**Legal Representative:**

- **HICKMAN Paul L (agent)**  
Oppenheimer Wolff & Donnelly, LLP, 38th Floor, 2029 Century Park East, Los Angeles, CA 90067-3024; US

	Country	Number	Kind	Date
Patent	WO	200108037	A2-A3	20010201
Application	WO	2000US20353		20000726
Priorities	US	99361338		19990726

**Designated States:** (Protection type is "Patent" unless otherwise stated - for applications prior to 2004)

AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY,  
CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI,  
GB, GE, GH, GM, HR, HU, ID, IL, IS, JP,  
KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT,  
LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ,  
PL, PT, RO, RU, SD, SE, SG, SI, SK, SL,  
TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU,  
ZW

[EP] AT; BE; CH; CY; DE; DK; ES; FI; FR; GB;



GR; IE; IT; LU; MC; NL; PT; SE;

[OA] BF; BJ; CF; CG; CI; CM; GA; GN; GW; ML;  
MR; NE; SN; TD; TG;

[AP] GH; GM; KE; LS; MW; MZ; SD; SL; SZ; TZ;  
UG; ZW;

[EA] AM; AZ; BY; KG; KZ; MD; RU; TJ; TM;

**Language** Publication Language: English

Filing Language: English

Fulltext word count: 86229

### Detailed Description:

...and client-side performance is improved. Unlike HTML, Java supports the notion of client-side validation, offloading appropriate processing onto the client for improved performance.

**Dynamic**, real-time Web pages can be created. Using the above-mentioned custom UI components, dynamic Web pages can also be created.

Sun's Java language... Improvements may be based on incremental operational refinements or through innovations, such new technologies. Improvements may typically be driven by the following activities.

Identifying and **approving** changes to the standard process definition on the basis of quantitative understanding of the process.

Providing adequate resources to effectively implement the approved changes in... the Service Desk's

assessment of incident and problem management?

Base Practice: 1 11 Receive Requests

Are requests handled immediately or do they require provisioning/**approval**? Does the Service Desk coordinate the **approval** of requests with the appropriate functions and notify requester of **approval**/rejection?

If request requires functions outside the Service Desk, how does the Service Desk pass responsibility to the appropriate personnel?

Do SLAs exist between the...ee service for the affected usersfor a specified amount of time.

BP Number 1 7

BP Name Agree to SLAs with users

BP Description Final **approval** should be obtained from the parties involved in the SLA agreement process.

This will be representatives from the IT organization, client management, and the users...document (eg. memo) describing it is

available.

1. 1.7 Agree to SLAs with users Prior to being finalized, SLAs were reviewed with users and their **approval** obtained. Signed off SLA document exists.

1. 1.8 Report on SLA performance Regular reporting on SLA performance occurs.

Reports include predefined metrics and KPIs...budgeting or costing that is used in SLA management?

Base Practice: 1.7 Agree to SLAs with Users

To what parties are SLAs submitted for **approval**?

How is **approval** of the SLA documented?

Where is information about the finalized SLA stored? Are SLA summaries available to users? Is there a system for users to...Are SLAs drafted based on the requirements identified?

Is a charge back structure defined and used to assign rewards/penalties based on SLA performance?

Is **approval** of the draft SLA obtained from all parties involved?

1. Are KPIs and metrics regularly reported to monitor SLA performance?

Process Area 1.1 SLA Management...the OLA with the vendor cannot equal or exceed 24 hours.

BP Number 1.6

BP Name Agree to OLAs with suppliers

BP Description Final **approval** should be obtained from the parties involved in the OLA agreement process.

This will include suppliers, information service personnel, and possibly representatives from the user...any provisions for rewards if OLA requirements are exceeded?

Base Practice: 1.6 Agree to OLAs with Suppliers

To what parties are OLAs submitted for **approval**?

How is **approval** of the OLA documented?

What is the approximate proportion of OLAs created with internal service providers compared to external suppliers?

Where is information about the...select service providers from the pool of candidates?

For each service provider, is an OLA prepared and are measures for assessing service levels specified?

Is **approval** of the OLA obtained from all parties involved in the OLA agreement process?

Are service levels measured and reported periodically to monitor OLA compliance?

Are...Service

Desk's assessment of incident and problem management?

Base Practice: 1.1.1 Receive Requests

Are requests handled immediately or do they require provisioning/**approval**? Does the Service Desk coordinate the **approval** of requests with the appropriate functions and notify

requester of **approval**/rejection?

If request requires functions outside the Service Desk, how does the Service Desk pass responsibility

to the appropriate personnel?

Do SLAs exist between the ...user representatives for review?

Does management or customer sponsor sign off plans?

What is the procedure for modification of the budgets or plans if initial **approval** is not obtained? Is the cost allocation plan reviewed to ensure that costs are accurately captured and prices are fair?

How does this occur?

Base...Change Control

Plan, Human Resources, Operations Personnel)

2. Are current procedures and resources periodically assessed with the intent to promote continuous improvement? What is the **approval** process for proposed solutions? Are these solutions evaluated

for impact?

79

. Are there regularly scheduled training programs that address User Administration procedures?

If

so, what...Description When ad hoc production orders come through many considerations, such as other processes running (backup/restoration, software distribution, etc.), need to be weighed before **approving** and sending the ad hoc order.

Example When ad hoc requests are considered procedures/requirements are used to modify the master computer schedule.

BP Number 2 5...feedback in Print management receives order to maintain feedback via, e-mail, reports and knowledge and experience meetings from customers and other process areas regarding issues, **approvals** and reviews.

Level 4 Assessment Indicators

Process Attribute Generic Practice Example of Assessment Indicator Assessment

Indicators

at Client

Process GP4.1 Establish Print management is...Print forms

1. Does the output/print management personnel review forms prior to their use throughout the distributed system? Reports? How is this process of **approval** managed (e.g. meetings, requests, sample stock, test runs, etc.)?
2. Can forms be collated and packed?
3. Are there certain printers on the network...established directory management and address servicing procedures?
2. Are current resources and procedures periodically assessed with the intent to promote continuous improvement? What is the **approval** process for proposed solutions? Are all potential stakeholders involved in the decision process? How often are these solutions implemented and

by whom? 3. How are...primarily to handle disasters rather than for restoring files accidentally deleted or overwritten. However, special requests for restoring individual files are processed through the Service Desk with prior **approval**.

After analyzing the request, the correct tape is determined and the corresponding directory files are restored. If necessary, changes are applied from the journal to the restored... Practice: 2 1

Define Security Objectives

What types of issues are covered by the formal security policy?

Was the security policy submitted to management for **approval**?

Is the security policy documented and available to customers and management?

Base Practice: 2 2 Develop security plan and policies

Please describe the contents of...errors.

Emergency, which require immediate attention and correction/inplementation.

PA's Base 3 1 Change Initiation

Practices 3 2 Change Impact Analysis/Assessment

145

Change **Approval**

3 4 Change Communication and Scheduling

3 5 Change Implementation Planning and Preparation

3 6 Change Request Tracking

3 7 Change Implementation

3 8 Change...service levels impacted by the proposed change. In addition, dependencies to other applications, databases or systems are identified.

BP Number 3 3

BP Name Change **Approval**

BP Description Authority to proceed with a Change Request can come from various sources depending on the type of change. The organization must be able... ..change request requires the Change Control Coordinator's or the Managing Directors signature before it will be implemented. It is up to

the person giving **approval** to clarify

any concerns regarding the change request with the

requestor (or the business leader, as appropriate). If necessary given the urgency of other...are recorded at time of

receipt.

3 2 Change Impact analysis template or an example of completed Analysis/Assessment analysis report.

3 3 Change **Approval** Example of completed change request form shows authorization by appropriate personnel.

3 4 Change Communication and A system for coordinating the planned date of a Scheduling... Process GP3.3: Plan for human Change requests are always handled

Resource resources proactively according to the documented policy (eg.

requests are not processed without appropriate **approval**).

GP3.4 Provide feedback in On completion of a change request, order to maintain knowledge feedback is solicited from the requester.

and experience

Level 4...consequences of the change impact analysis (i.e. is the change request rejected if the change analysis yields particular results)?

Base Practice: 3.3 Change **Approval**

Whose **approval** is needed before a change request can be implemented? Does the person(s) whose **approval** is necessary depend on the scope or priority level of the change?

How is **approval** obtained and documented?

Is the change requestor notified of change **approval** or rejection?

Base Practice: 3.4 Change Communication and Scheduling

Once **approval** is obtained, what is the process for estimating the time and scheduling the change? Are other completion times and dates factored into the estimated... submitted?

Prior to implementing a proposed change, is an impact analysis performed to identify the effects of the proposed change?

Does a change request require **approval** from designated personnel before being implemented?

Are the implementation time and completion date estimated and scheduled?

Are all affected parties notified of a scheduled change...taking place. If changes to the schedule are required, Deployment is responsible for coordinating the changes across all of the groups involved and seek management **approval** for the changes.

PA's Base Confirm schedule with all key groups periodically

Practices Determine whether schedule will be impacted based on issues or problems...and external groups may result in a scheduling conflict or a need for longer lead time. The deployment schedule should accommodate this issue and managerial **approval** should be gained.

Example If an external vendor cannot complete hardware installations prior to scheduled deployment, the deployment schedule should be changed accordingly and management should **approve** prior to any advancement.

BP Number 3.4

BP Name Report on progress of deployment plan

BP Description The current status of deployment and any...Assessment Indicator Assessment Indicators

at Client

3.1 Confirm schedule with all Feedback is received during a meeting with payroll key groups periodically noting the **approval** of a May deployment, as the new

date is after the tax season. An electronic or hard copy

schedule noting prep, training and customer types....5 Maintain Feedback might be collected

via meetings

communication among team and reports from physical planning and members management regarding lead times. GP2.7 Employ version **Approvals** are gained on schedules to control to manage changes to accommodate the latest issues noted on Level 3 Assessment Indicators Process Attribute Generic Practice Example...

8/3K/9 (Item 6 from file: 349)

DIALOG(R)File 349: PCT FULLTEXT

(c) 2011 WIPO/Thomson. All rights reserved.

00775305

**A SYSTEM, METHOD AND COMPUTER PROGRAM FOR DETERMINING  
CAPABILITY LEVEL OF PROCESSES TO EVALUATE OPERATIONAL MATURITY  
IN AN ADMINISTRATION PROCESS AREA**

SYSTEME, PROCEDE ET ARTICLE MANUFACTURE DE VERIFICATION D'UN  
PROCESSUS A MATURETE OPERATIONNELLE PAR DETERMINATION DU NIVEAU  
D'APTITUDE DANS UN DOMAINE DE PROCESSUS TRAITEMENT  
D'ADMINISTRATION UTILISATEUR

**Patent Applicant/Patent Assignee:**

- **ACCENTURE LLP**

1661 Page Mill Road, Palo Alto, CA 94304; US; US(Residence); US(Nationality); (For all designated states except: US)

**Patent Applicant/Inventor:**

- **GREENBERG Nancy S**

5529 Newton Avenue South, Minneapolis, MN 55410; US; US(Residence); US(Nationality); (Designated only for: US)

- **WINN Colleen R**

11472 Fairfield Road #103, Minnetonka, MN 55305; US; US(Residence); US(Nationality); (Designated only for: US)

**Legal Representative:**

- **HICKMAN Paul L (agent)**

Oppenheimer Wolff & Donnelly, LLP, 1400 Page Mill Road, Palo Alto, CA 94304; US

	Country	Number	Kind	Date
Patent	WO	200108035	A2-A3	20010201

	Country	Number	Kind	Date
Application	WO	2000US20238		20000726
Priorities	US	99360928		19990726

**Designated States:** (Protection type is "Patent" unless otherwise stated - for applications prior to 2004)

AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG,  
BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE,  
DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH,  
GM, HR, HU, ID, IL, IN, IS, JP, KE, KG,  
KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV,  
MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ,  
PL, PT, RO, RU, SD, SE, SG, SI, SK, SL,  
TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN,  
YU, ZA, ZW

[EP] AT; BE; CH; CY; DE; DK; ES; FI; FR; GB;  
GR; IE; IT; LU; MC; NL; PT; SE;

[OA] BF; BJ; CF; CG; CI; CM; GA; GN; GW; ML;  
MR; NE; SN; TD; TG;

[AP] GH; GM; KE; LS; MW; MZ; SD; SL; SZ; TZ;  
UG; ZW;

[EA] AM; AZ; BY; KG; KZ; MD; RU; TJ; TM;

**Language** Publication Language: English

Filing Language: English

Fulltext word count: 86405

### Detailed Description:

...validation, offloading appropriate processing onto the client for improved performance. Dynamic, real-time Web pages can be created. Using the above-mentioned custom UI components, **dynamic** Web pages can also be created.

Sun's Java language has emerged as an industry-recognized language for "programming the Internet." Sun defines Java as...Improvements may be based on incremental operational refinements or through innovations, such new technologies. Improvements may typically be driven by the following activities.

Identifying and **improving** changes to the standard process definition on the basis of quantitative understanding of the process.

9 Providing adequate resources to effectively implement the approved changes...the Service Desk's

assessment of incident and problem management?

Base Practice: 1 11 Receive Requests

Are requests handled immediately or do they require provisioning/**approval**? Does the Service Desk coordinate the **approval** of requests with the appropriate functions and notify requester of **approval/rejection**?

If request requires functions outside the Service Desk, how does the Service Desk pass responsibility to the appropriate personnel?

Do SLAs exist between the...be discounted or free service for the affected users for a specified amount of time.

BP

:BP Name Agree to SLAs with users

BP Description- Final **approval** should be obtained from the parties involved in the SLA agreement process.

This will be representatives from the IT organization, client management, and the users...a document (eg. memo) describing it is available.

1 7 Agree to SLAs with users Prior to being finalized, SLAs were reviewed with users and their **approval** obtained. Signed off SLA document exists.

1. 1. 8 Report on SLA performance Regular reporting on SLA performance occurs.

I Reports include predefined metrics and AP...budgeting or costing that is used in SLA management?

Base Practice: 1 7 Agree to SLAs with Users

To what parties are SLAs submitted for **approval**?

How is **approval** of the SLA documented?

Where is information about the finalized SLA stored? Are SLA summaries available to users? Is there a system for user@ to...Are SLAs drafted based on the requirements identified?

Is a charge back structure defined and used to assign rewards/penalties based on SLA performance?

Is **approval** of the draft SLA obtained from all parties involved?

Are KPIs and metrics regularly reported to monitor SLA performance?

Process Area 1. 1 SLA Management...the OLA with the vendor cannot equal or exceed 24 hours.

BP Number 1 6

BP Name Agree to OLAs with suppliers

BP Description Final **approval** should be obtained from the parties involved in the OLA agreement process.



This will include suppliers, information service personnel, and possibly representatives from the user...any provisions for rewards if OLA requirements are exceeded?

Base Practice: 1 6 Agree to OLAs with Suppliers

To what parties are OLAs submitted for **approval**?

How is **approval** of the OLA documented?

What is the approximate proportion of OLAs created with internal service providers compared to external suppliers?

Where is information about the...select service providers from the pool of candidates?

For each service provider, is an OLA prepared and are measures for  
- assessing service levels specified?

Is **approval** of the OLA obtained from all parties involved in the OLA agreement process?

Are service levels measured and reported periodically to monitor OLA  
- compliance?

Are...Service

Desk's assessment of incident and problem management?

Base Practice: 1 1 1 Receive Requests

Are requests handled immediately or do they require provisioning/**approval**? Does the Service Desk coordinate the **approval** of requests with the appropriate functions and notify requester of **approval**/rejection?

If request requires functions outside the Service Desk, how does the Service Desk pass responsibility to the appropriate ...user representatives for review?

Does management or customer sponsor sign off plans?

What is the procedure for modification of the budgets or plans if initial **approval** is not obtained? Is the cost allocation plan reviewed to ensure that costs are accurately captured and prices are fair?

How does this occur?

Base...Change Control

Plan, Human Resources, Operations Personnel)

2. Are current procedures and resources periodically assessed with the intent to promote continuous improvement? What is the **approval** process for proposed solutions? Are these solutions evaluated

for impact?

3. Are there regularly scheduled training programs that address User Administration procedures? If

so, what...scription When ad hoc production orders come through many considerations, such as other processes running (backup/restoration, software distribution, etc.), need to be weighed before **approving** and sending the ad hoc order.

Ejea@nj lde' When ad hoc requests are consideredproceduresrequirements are used to modify the master computer schedule.

BP Number...feedback in Print management receives order to maintain feedback via, e-mail, reports and

knowledge and experience meetings from customers and other process areas regarding issues, **approvals** and reviews.

#### Level 4 Assessment Indicators

Process Attribute Generic Practice Example of Assessment Indicator Assessment Indicators

at Client

Process GP4.1 Establish Print management is...Print forms

1. Does the output/print management personnel review forms prior to their use throughout the distributed system? Reports? How is this process of **approval** managed (e.g. meetings, requests, sample stock, test runs, etc.)?

... management and address service processes.

2. Are current resources and procedures periodically assessed with the intent to promote continuous improvement? What is the **approval** process for proposed solutions? Are all potential stakeholders involved in the decision process? How often are these solutions implemented and by whom? (10).

. How...to handle disasters

101

rather than for restoring files accidentally deleted or overwritten. However, special requests for restoring individual files are processed through the Service Desk with prior **approval**.

After analyzing the request, the correct tape is determined and the corresponding directory files are restored. If necessary, changes are applied from the journal to the restored copy...Practice: 2.1 Define Security Objectives

What types of issues are covered by the formal security policy?

Was the security policy submitted to management for **approval**?

Is the security policy documented and available to customers and management?

Base Practice: 2.2 Develop security plan and policies

Please describe the contents of...Emergency, which require immediate attention and correction/implementation.

PA's, Base 3.1 Change Initiation

Practices 3.2 Change Impact Analysis/Assessment

3.3 Change **Approval**

ge Communication and Scheduling

3.4 Change

3.5 Change Implementation Planning and Preparation

3.6 Change Request Tracking

3.7 Change Implementation

3.8...service levels impacted by the proposed change. In addition, dependencies to other applications, databases or systems are identified.

BP Number 3.3

## BP Name Change **Approval**

BP: Description Authority to proceed with a Change Request can come from various sources depending on the type of change. The organization must be able ... request requires the Change Control Coordinator's or the Managing Director's signature before it will be implemented. It is up to the person giving **approval** to clarify any concerns regarding the change request with the requestor (or the business leader, as appropriate). If necessary given the urgency of other is...are recorded at time of receipt.

3 2 Change Impact Impact analysis template or an example of completed Analysis/Assessment analysis report.

3 3 Change **Approval** Example of completed change request form shows authorization by appropriate personnel.

3 4 Change Communication and A system for coordinating the planned date of a Scheduling...  
...repeatedly  
Process GP3.3: Plan for human Change requests are always handled  
Resource resources proactively according to the documented policy (eg.

requests are not processed without  
appropriate **approval**).

GP3.4 Provide feedback in On completion of a change request,  
order to maintain knowledge feedback is solicited from the requester.

and experience

Level 4 Assessment...consequences of the change impact analysis (i.e. is the change request rejected if the change analysis yields particular results)?

Base Practice: 3 3 Change **Approval**

Whose **approval** is needed before a change request can be implemented? Does the person(s) whose **approval** is necessary depend on the scope or priority level of the change?

How is **approval** obtained and documented?

Is the change requestor notified of change **approval** or rejection?

Base Practice: 3 4 Change Communication and Scheduling

Once **approval** is obtained, what is the process for estimating the time and scheduling the change? Are other completion times and dates factored into the estimated time ...submitted?

Prior to implementing a proposed change, is an impact analysis performed to identify the effects of the proposed change?

Does a change request require **approval** from designated personnel before being implemented?

Are the implementation time and completion date estimated and scheduled?

...taking place. If changes to the schedule are required, Deployment is responsible for coordinating the changes across all of the groups involved and seek management **approval** for the changes.

PA@s @Base Confirm schedule with all key groups periodically  
Practices Determine whether schedule will be impacted based on issues or problems...and  
external groups may result in a scheduling conflict or a need for longer lead time. The  
deployment schedule should accommodate this issue and erial **approval** should be gained.

manag

Example If an external vendor cannot complete hardware installations prior to scheduled  
deployment, the deployment schedule should be changed accordingly and management should  
**approve** prior to any advancement.

AO Number 3 4

.-BP Name Report on progress of deployment plan

-BP Description The current status of deployment and any... ..Assessment Indicator Assessment  
Indicators

at Client

3 1 Confirm schedule with all Feedback is received during a meeting with payroll key groups  
periodically noting the **approval** of a May deployment, as the new  
date is after the tax season. An electronic or hard copy  
schedule noting prep, training and customer types...GP2.5 Maintain Feedback might be collected  
via meetings  
communication among team and reports from physical planning and  
members management regarding lead times. GP2.7 Employ version **Approvals** are gained on  
schedules to control to manage changes to accommodate the latest issues noted on Level 3  
Assessment Indicators Process Attribute Generic Practice Example...

8/3K/10 (Item 7 from file: 349)

DIALOG(R)File 349: PCT FULLTEXT

(c) 2011 WIPO/Thomson. All rights reserved.

00761431

**A SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR PROVIDING  
COMMERCE-RELATED WEB APPLICATION SERVICES**

SYSTEME, PROCEDE ET ARTICLE MANUFACTURE DESTINES A LA FOURNITURE  
DE SERVICES D'APPLICATION DANS LE WEB LIES AU COMMERCE

**Patent Applicant/Patent Assignee:**

- **ACCENTURE LLP**  
100 South Wacker Drive, Chicago, IL 60606; US; US(Residence); US(Nationality)

**Inventor(s):**

- **GUHEEN Michael F**  
2218 Mar East Street, Tiburon, CA 94920; US
- **MITCHELL James D**  
3004 Alma, Manhattan Beach, CA 90266; US
- **BARRESE James J**  
757 Pine Avenue, San Jose, CA 95125; US

**Legal Representative:**

- **BRUESS Steven C (agent)**  
Merchant & Gould P.C., P.O. Box 2903, Minneapolis, MN 55402-0903; US

	Country	Number	Kind	Date
Patent	WO	200073957	A2-A3	<b>20001207</b>
Application	WO	2000US14420		20000525
Priorities	US	99321492		19990527

**Designated States:** (Protection type is "Patent" unless otherwise stated - for applications prior to 2004)

AE, AG, AL, AM, AT, AT (utility model), AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, CZ (utility model), DE, DE (utility model), DK, DK (utility model), DM, DZ, EE, EE (utility model), ES, FI, FI (utility model), GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KR (utility model), KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (utility model), SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW

[EP] AT; BE; CH; CY; DE; DK; ES; FI; FR; GB; GR; IE; IT; LU; MC; NL; PT; SE;

[OA] BF; BJ; CF; CG; CI; CM; GA; GN; GW; ML; MR; NE; SN; TD; TG;

[AP] GH; GM; KE; LS; MW; MZ; SD; SL; SZ; TZ; UG; ZW;

[EA] AM; AZ; BY; KG; KZ; MD; RU; TJ; TM;

**Language** Publication Language: English

Filing Language: English

Fulltext word count: 150171

**Detailed Description:**

...Development ana.

1.13 including the following.

Testing Tools Development Tools.

EmbeddedJava Application Environment

JavaBeans Development Kit

JavaBlend

Java Compiler Compiler

Java Development Kit

Java **Dynamic** Management Kit (JDMK)

JavaHelp

Java Management AN (JMAPI)

Java JIT Compiler

Java SDK

Java WorkShop

NEOWorks

Personal Java Application Environment

Serviet Development Kit

Product6 ASN... Management unit needs to.

0 Ensure all security issues are effectively addressed throughout the program (all business and IT processes).

0 Act as facilitator and **approving** body for all new and existing initiatives that contain security components.

40

Own responsibility for the organization and facilitation of working groups that would address ...increased productivity, decreased lead times, or lower error rates. If the business case is not testable, the benefits realization test becomes more of a buyer **signoff**.

Ideally, benefits realization test occurs prior to complete deployment of the system and utilizes the same environment that was used for the service-level test... ..increased productivity, decreased lead times, or lower error rates. If the business case is not testable, the benefits realization test becomes more of a buyer **signoff**.

g) Are quality requirements being tested?

h) Are technical requirements being tested?

i) Are junctional user requirements being tested?

The following is an overview of the product...required for the Problem Management application. This is closely tied with the Configuration Management tools. Only one person should have the rights to review and **approve** problem analysis tasks as well as problem migration activities.

Implementation Considerations

- a) How are problems handled at each stage?
- b) How

#### **Claims:**

...Placement (outbound sm17;19j)GS Communifies of Interes Capsb Order T1  
 Calculations 7 Delivery (intitourictinjShopping can] F Discussion, Match Web c;nteni I Content  
**Approval**Tax &amp; Shippin9 Small) Sp Pr ,Os(newignou S) acific userCompete Productsj  
 NUTS: r 'o . m 'Servic @"" I Customer Feed Content Worklowes (ph )Si...THE  
 DATABASEFigure 23B2310DEVELOPING CONTENT OF A DATA INTERFACE FOR  
 ACCESSING DATA ON A 2400NETWORKMANAGING THE CONTENT OF THE DATA  
 INTERFACE**APPROVING** THE PUBLICATION OF THE CONTENT BEFORE  
 TRANSMISSION OF THE 2404CONTENT2406TESTING THE CONTENT OF THE DATA  
 INTERFACE1408Figure 24GENERATING A CURRICULUM...

8/3K/11 (Item 8 from file: 349)  
 DIALOG(R)File 349: PCT FULLTEXT  
 (c) 2011 WIPO/Thomson. All rights reserved.

00459180

#### **OPERATING RESOURCE MANAGEMENT SYSTEM SYSTEME DE GESTION DE RESSOURCES DE FONCTIONNEMENT**

#### **Patent Applicant/Patent Assignee:**

- **ARIBA TECHNOLOGIES INC**
- **ADAMS Norman**
- **BROWN Marc**
- **CARLSTROM Brian**
- **ELKIN Brian**
- **HEGARTY Paul**
- **HASKIN Guy**
- **PUTANEC Boris**

#### **Inventor(s):**

- **ADAMS Norman**
- **BROWN Marc**
- **CARLSTROM Brian**
- **ELKIN Brian**
- **HEGARTY Paul**

- HASKIN Guy
- PUTANEC Boris

	Country	Number	Kind	Date
Patent	WO	9849644	A1	19981105
Application	WO	98US8407		19980427
Priorities	US	9744372		19970428

**Designated States:** (Protection type is "Patent" unless otherwise stated - for applications prior to 2004)

AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY,  
 CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI,  
 GB, GE, GH, HU, IL, IS, JP, KE, KG, KP,  
 KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD,  
 MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO,  
 RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR,  
 TT, UA, UG, US, UZ, VN, YU, ZW, AT, BE,  
 CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,  
 IT, LU, MC, NL, PT, SE

**Language** Publication Language: English

Fulltext word count: 13618

### English Abstract:

...module generates the requisition record responsive to a combination of input by a requestor and operating resource information in an operating resource information database. An **approval** path determining module, responsive to the requisition record and to **approval** rules in an **approval** rules database, determines an **approval** path for the requisition record, among various ones of a plurality of possible **approvers**, required to **approve** the requisition record. An **approval** path handling module guides the requisition record along the determined **approval** path, and the **approval** path handling module generates a global **approval** indication in response to the requisition record successfully traversing the **approval** path.

### Detailed Description:

...requisitioning process.

Use of the system need not be limited to a select purchasing group.

Ubiquitous and Easy Information Access for All Employees - Requesters and **approvers** alike can see the current state of any of their requisitions at any time and, thus, are always kept in the loop when something changes about their requisition.



M Authenticated **Approval** Flow -- The system enforces the corporation's business rules and validates requisitions, ensuring accurate and complete data.

M Adapters For Integrating with the Enterprise -- The... ..via the web browser), on every desktop (shown in Fig. 1 as "Mae", "Win95", "WinN7 and "Unix"), and provides the user interface for creating and **approving** requisitions. The Java server software 102 preferably runs on a single shared machine, and provides "back-end" services.

Supplementing the Enterprise Commerce Server 102 are in a typical embodiment, from requisition creation to **approval**, to receipt of requisitioned goods/services, and to reconciliation. In Fig. 3, the reference numeral 302 designates process steps associated with creating a requisition.

Figure... ..a supplier and part explicitly (310). Entering such ad-hoc items, items that are not in the list of approved items, will typically trigger new **approval** rules. For example, the **approval** rules of many companies will cause the Purchasing Department to be put into the **approval** loop at this point, to require a Purchasing Agent to decide whether to **approve** the new item. Because ad-hoc entry usually involves additional overhead, the wizard guides the employee through the process in such a way as to... ..than typing, on channeling the employee toward standard answers, and on generating error-free requisitions.

Any employee who handles a requisition, be it requester or **approver**, can add commentary or attach documents to the requisition. The ability to comment and explain can go a long way toward maintaining alignment, making requisitions understandable to **approvers**, allow **approvers** to provide feedback to requesters, and help **approvers** make a decision about whether to **approve** the request.

After a request is submitted, another piece of user interface software 500 comes into play: the Organizer 504 (Figure 5). In a preferred... ..504 software provides a folders-style view of existing requisitions, designed to help group and organize large collections of requisitions.

When a request is submitted, **approval** software ( **approval** engine 110 in Figure 1; step 322 in Figure 3; **approval** flow software 602 of the system environment 404, in Figure 6) inspects the **approval** rules of the company, decides who needs to **approve** the request, and notifies the first required **approvers**, preferably by e-mail, that there is a requisition waiting for their attention. In one embodiment, the e-mail notification message includes a URL hyperlink that points the **approver** directly to the Organizer software 504 via his browser, to display the requisitions waiting for this person's **approval**. The **approver** can **approve** or deny, and make comments, asking for more information or clarification.

Whenever there is a status change in a requisition, notification software 120 sends an e-mail message notifying the requester and any other interested parties. The system uses notification e-mails throughout the **approval** process to keep users informed about the current state of a requisition. Requesters can also use the Organizer software 504 to check the status of... ..or has

not approved it, when it has been fully approved, and so on.

Using the Organizer 504 and the commenting mechanism, everyone in the **approval** process (e.g., **approvers**, requisitioners, and Purchasing Agents) can ask each other questions, view the status of a requisition, or make comments about the requisition, reducing confusion and improving... ..Figure 1) (see also, Fig. 7).

There are three ordering modules 702 (see Figure 7): a Purchasing Card module, a Direct Order module, and a **Purchase Order** module.

Eventually the requisition will be approved, submitted, and fulfilled. As discussed above, the system may communicate orders to the supplier via conventional means (fax addition to the requisitioning population -- the requesters and **approvers** -- there is another class of users: members of the Purchasing Department. The Purchasing Agents are responsible for the buying practices of the company, ensuring that... ..the gain from automation. This information can be a valuable tool for gaining understanding and using that understanding to make improvements, such as modifying the **approval** processes or switching suppliers or updating the history of purchases to encourage different buying patterns by end-users in the future.

The system environment 404... ..fields can be used. For example, a company might wish to extend the set of data fields to describe its own accounting policies and categories.

**Approval** rules are the conditions that a company uses to decide which **approvers** are required for a particular requisition. The system preferably provides a mechanism for describing the **approval** rules that is flexible enough to model ...will have its own set of rules, although there are often basic similarities, and many rules can be copied from simple examples. For example, an **approval** rule may be expressed as a set of conditional expressions, such as "If the amount of this purchase is over \$25,000 and it is for software, then the Information Systems department must **approve** the purchase. "

There are at least two things to note about the **approval** rules. First, **approval** rules can be based on any field in a requisition, including the fields that are added during the implementation process. So, in addition to standard **approvals** based on requisition or line item amounts, for example, the Facilities Manager might need to **approve** any furniture purchase, or the IS department might have to **approve** any computer system purchase.

In addition, an **approver** designation does not have to be given to a particular individual in the company. Rather, a particular role can be designated in an **approval** rule as an **approver**. An example role is the "CFO" role. At any given time, this role is played by a single individual in the company, but if there is a new CFO hired, then all the requisitions that are awaiting **approval** by the CFO can be approved by the new CFO when he comes on board, without any system maintenance. When the individual who is the new CFO is designated in the system as

CFO, he will be notified of all requisitions pending **approval** for the CFO role.

Roles can also describe a group of people. For example, there is the role of Purchasing Agent. There might be any... ..company, but if the role Purchasing Agent is assigned to a requisition, then all individuals designated by the Purchasing Agent role see it for their **approval**.

- 12 In one embodiment, if any one of them approves it, the requisition is approved for that role.

Adapters 122 (Fig. 1) and 800 (Fig... ..etc.). The ERP adapters can obtain simple information from the ERP like units of measure, accounting information, etc., as well as item templates, supplier information, **approval** matrices, and other relatively static information. They are also capable of storing the entire approved ...for someone else. That is, allow the creator and submitter to be different people. If the requester and the submitter are different, then the standard **approval** rules will put the requester as the first **approver**.

e. Allow the employee to specify a hold date on a requisition. The hold date is the date that the employee would like the requisition...it in), or leave it out, to be chosen by the Purchasing Agent. Requisitions for items that are not from approved sources typically trigger special **approval** rules, such as requiring a Purchasing Agent to **approve** the new item and supplier. The system provides facility for each company to define its own rules for handling such requests.

- 18

4 Fill in...Quantity X Price. Derived  
Extensible fields, custom to this company.

- 19

#### 5. Comments and attachments

Any employee who handles a requisition, be it requester or **approver**, can add commentary or attach documents to the requisition, helping everyone who sees it to better understand the requisition. The ability to comment and explain can go a long way toward making requisitions understandable to **approvers**, allowing them to provide feedback to requesters, and help them make a decision about whether to **approve** the request.

The commenting mechanism.

a. Allows users to add textual comments to any requisition or line item, using "threading" to maintain context.

b. Allows users to specify the audience for a comment, which can be any of **Approvers**, Requesters, Suppliers, Purchasing, or All. Comments are visible only to the specified audience.

C. Allows users to attach electronic documents to comments. To ensure platform... ..employee has finished filling out a requisition and asked to submit it, the system will perform the following checks before actually submitting the requisition for **approval**.

a. Find all mandatory fields (as distinguished from optional ones), and ensure that they have values. If there are any missing values, then the requisition date and time, as the submission date of the requisition.

f. Determine the **approval** path for this requisition, using the **approval** rules defined in the business rules for the company, and allow the employee to preview the **approval** path. Allow the employee to either confirm the submission, or cancel it and return to editing the requisition.

#### The Organizer

The user interface software for categorizing and classifying requisitions is known as the Organizer 504 (Figure 5). **Approvers** use the Organizer software to **approve** or deny requisitions and requesters use it to check status and history.

When a request is submitted, the system checks the **approval** rules of the company, decides which users need to **approve** the request, and in what order, - 21 and then notifies the first **approver** that there is a requisition waiting for attention.

Each **approver** sees new requisitions in a folder of incoming requisitions, and will need to take action on the requisition to move it to a different folder.

#### 1 . **Approving** or Denying Requisitions

When an **approver** goes to the Organizer interface, be it from a notification message, a bookmark, or some other hyperlink, the Organizer displays the incoming requisitions for that **approver**, showing the information in Table 3, below, for each requisition.

Table 3: Fields of an **approval** request

Field T Explanation

1. Role Role required for this **approval**, such as "CFO".
2. Reason The reason this **approver** needs to **approve**; this is the justification field
3. Actual **Approver** The name of the person who is filling the **approval** role.

This is typically the **approver's** name, if the **approver** is looking at incoming requisitions.

4. Required/Optional Boolean indicating whether this **approval** is required, or whether this **approver** is a "watcher".

## 5. Submission Date The submission timestamp.

Whenever an **approver** acts on a requisition, the system timestamps the requisition with the name of the **approver** and the time of the action.

If an **approval** is marked as required, the **approver** can take any of the following actions on the requisition.

a. **Approve** the requisition. An **approval** will trigger any notifications specified in the business rules for this company, mark the request as approved for this **approver**, and add the request to the incoming folder for the next **approver** in the **approval** chain. After **approving** a request, the **approver** can move it into some other folder, or leave it in the incoming folder.

- 22

b. **Deny** the requisition. When an **approver** denies a requisition, the system sends an e-mail notification to the requester, and stops any further **approval** requests in this serial **approval** chain. If the requester does nothing in response to a notification of denial, the request will eventually time out. If the requester modifies the request and resubmits it, the system starts the **approval** process again, as described in step 5) below.

c. Add an additional **approver** to the **approval** chain, either before or after this **approval**. For example, an **approver** might want to say "Please ask Ed if he approves, and then come back to me".

d. Add comments.

e. **Modify** the requisition. Not all **approvers** can change all fields, however: a Purchasing Agent can modify any field of a requisition; other **approvers** can modify only a limited set of fields in the requisition. The definition of which fields **approvers** can modify is part of the company's configuration of the data fields and is typically set up during installation.

When an **approver** modifies any field of a requisition, the system **recalculates** the required **approvals** and invalidates any existing **approvals** for that line item (if it was a line item that changed) or for the entire requisition (if the requisition itself was changed). Modifying a field can thus trigger reapprovals from users who have already approved the requisition, or trigger the addition of new **approvers** into the chain, depending on the **approval** rules.

If the **approver** is marked as Optional, then this **approver** is a watcher, not a true **approver**. Watchers are bystanders: they see the requisition but their **approval** is not required. Watchers can take any of the following actions on the requisition.

- 23

Add an additional **approver** to the **approval** chain, either before or after this **approval**.

0 Add comments.

## 2. **Approving** in the place of others

The system maintains the notion of chain of command derived from the "immediate supervisor" information in each employee's personal profile. Using that information, the system allows certain authorized **approvers** to **approve** in the place of another **approver**.

a. The system allows **approvers** to get a list of the requisitions that are waiting for **approval** from a lower-level **approver** (as defined by the business rules) and **approve** them directly. A high-level **approver** can explicitly **approve** in the place of any lower-level **approver** if the two **approvers** are in the same chain of command.

## 3. Removing Requisitions or **Approvals**

a. A requester can withdraw his or her own requisitions at any point during the **approval** process, until the requisition is fully approved. A withdrawn request returns to the Unsubmitted state and any **approvals** that have been recorded so far will be removed.

b. An employee who has the role of Purchasing Agent can remove **approvals** from any requisition.

## 4. Organizing requisitions

The Organizer helps employees organize groups of requisitions. It allows employees to.

- 24

a. Sort the requisitions by any... ..employee can use the value of that field to restrict the information being displayed.

C. View the details of any requisition, including all line items, **approvals**, and comments.

d. Put the results of a search into a folder. For example, a purchasing agent might wish to examine all outstanding requisitions for...specified fields of their own personal profiles, in a form consistent with the rest of the UI.

b. Submits all changes to personal profiles for **approvals**, as described in the **approval** rules of the company.

C. Allows employees to view the Human Resources data fields that are passed through from the HRMS adapter.

d. Allows employees... ..Profile

Field Name Explanation Intrinsic?

1. Organizational Level, Numeric degree of separation from CEO. Intrinsic 2. Delegation of authority (DOA) Any employee can designate **approval** authority to Intrinsic another user, for some period of time.
3. Start date of DOA Start date for DOA. Intrinsic
4. Termination date of DOA... ...5: Fields of a System Profile
1. System Name Name of the company. Intrinsic
2. URL URL of home page for this system Intrinsic
3. **Approval** escalation time Default interval before **approval** is escalated. Intrinsic 4. Time-out interval Time span before a requisition times out, if it has been in Intrinsic the system with no action...arises most often when there is a requisition for an item that is not in the Product Information Database. If the Purchasing Agent decides to **approve** the item, he or she will create a new item template for it and decide whether to add it to the Product Information Database.

b...item. Extrinsic

17. List unit price Purchase price, per unit, set by supplier Extrinsic
18. Buyer Role responsible for buying the part; input to the **approval** Extrinsic rules
19. Taxable Boolean indicating whether item is taxable Extrinsic
20. Supplier Part Number ID for this item, from the supplier Extrinsic
21. Manufacturer ... ..buying patterns (e.g., are we buying too much of something or too little?), to reports on the process itself (e.g., who is not **approving** in a timely manner). This information can help the company refine its practices, say by modifying the **approval** processes or switching suppliers.

- 29

1. Defining Rep.Qrts  
The system provides a variety of reports to categorize and group the information contained in the... ..a period I(High)
3. Requisitions still to be approved, by whom I(High)
4. Line items by supplier I(High)
5. Line items by **approver** I(High)
6. Average # of lines I(High)
7. Requisitions by commodity 2 (Medium)
8. Average time to **approve** 2 (Medium)
9. Requisitions denied, grouped by whom 3 (Low)

- 30

- 3 -Standard Reports available to purchasing agents  
Provide the standard reports as shown in...9. Unreceived orders by 3 (Medium)  
Supplier
10. Number of Watching for people who consistently order just 3 (Low)  
requisitions initiated by a under an **approval** limit  
given employee, in some  
period
11. Paper vs. electronic Number of electronic requests submitted, as 3 (Low)

compared to number of paper ones

12. % of items ordered Tracking ad-hoc items vs. catalog items 3 (Low)  
that were ad-hoc

System Environment

### Approval Flow

Each company generally has its own **approval** process for defining who has to **approve** each requisition. The system models this process with a set of **approval** rules, which each company can parameterize and extend. The **approval** - 31 rules are defined as part of the installation process, but can be modified by the customer's system administrator at any time.

The **approval** rules are preferably stored in text files that can be edited with any text file editor.

### 1 . Parameterizing **Approval** Rules

The simplest form of **approval** rules is a tabular file format, which describes values to be used in the rules. This file format allows the customer to.

a. Parameterize the **approval** rules by editing the values in the to tabular file. For example, a company can change the dollar amounts to be associated with **approval** by various management levels, without changing the **approval** rule itself.

b. Change the parameters while the system is running. The system will read in new parameters without downtime.

### 2. **Approval** Rules

For describing the **approval** rules, the system provides a simple scripting language, generally flexible enough to describe any condition or set of conditions file **approval**. In one embodiment, each rule has.

a. A justification field, to be used as explanation for why the rule was invoked.

b. A predicate, which... ..fields that this particular company has added.

- 32

C. A consequent, for when the predicate applies. The consequent designates which role or roles need to **approve** the requisition. For example, a company might write a rule that requires employees with the role of purchasing agent to **approve** any requests that are for amounts over \$200 and that have a ship-to ...amount, the \$200, will be specified in the tabular file; the predicate-consequent will be in the rules file.

d. A way to describe which **approvals** can be done serially, and which can be done in parallel. For example, an organization may want the management chain **approvals** to go serially, but other **approvals** (like Facilities and IS) to go in parallel.

### 3. Buyer Assignment Rules



Each line item in a requisition has an assigned Purchasing Agent. The system sets the assigned Purchasing Agent before submitting the request for **approval**. Each company can define its own rules for how buyers are assigned, using the same mechanism used for defining **approval** rules. For example, a company might wish to have the assignment of buyer be dependent both on the type of the commodity and the amount of the purchase.

#### 4. Escalation and Timing Out

The system provides the ability to escalate an **approval**, either manually or automatically, for occasions when an **approver** has not responded to a request for **approval**. Escalating an **approval** request moves it up the management chain, to the **approver's** immediate supervisor.

The system provides the following features for escalation.

- 33 a. A requester can escalate a request manually.

b. If an **approver** has not responded to a request for **approval** within the escalation time period defined in the system profile, the system will escalate the **approval** request automatically. Escalation will continue up the chain as necessary, until someone takes action or there is an employee with no supervisor.

C. If a... ..any request that has been submitted but not yet fully approved within the specified time frame will be escalated to the administrator.

d. Once an **approval** request has been escalated, the original designated **approver** can no longer take action on that request.

#### 5. Delegation of Authority

Delegation of authority (DOA) is a substitution of one **approver** for another in a specified time period, say when an **approver** is on vacation. In one embodiment, the system supports delegation of authority in the following ways.

a. Any employee can delegate his or her authority... ..in effect. The DOA is stored in the employee's personal profile: like all changes to personal profiles, delegations of authority are subject to the **approval** rules of the company. An employee cannot delegate to more than one person at a time, or split the DOA among more than one designee... ..is a delegation of authority for an employee, and the date for the delegation has not expired, then the system will allow the delegate to **approve** in the place of the employee.

C. Log all delegations of authority as part of ...Authentication  
Role Functionality

1. System Administrator nDesignate other employees as administrators or purchasing agents  
nLoad new business rules into the server
2. Purchasing Agent wRemove **approvals** from requisitions

mEdit any field of a requisition

mModify the Product Information Database

EModify the specific fields of the supplier database

2. Events and Notification ... ..it is possible to turn off notification altogether is a decision made on a per-company basis.

Table 10: Events Requiring Notification

Event Action

1. **Approval** is now required Notify activated **approver**.

2. An **approver** takes action: approves or denies. Notify requester.

3. New **approver** or watcher added Notify requester.

4. Requisition has been modified Notify requester.

5. Final receipt submitted Notify Purchasing.

6. PO# Assigned to Requisition Line Item... ..Notify requester that a receipt is passed and no receipt acknowledgment has been sent. required.

8. Requisition has been escalated to next-level Notify current **approver** and **approver**. activated **approver** and requester.

9. Requisition is soon to be escalated. Notify **approver**.

10. Requisition is soon to time out Notify requester.

11. Requisition has timed out Notify requester.

3. Database Support

The system uses a database...and submits it for fulfillment. When a requisition has been fully approved, the system will.

Timestamp it with the date and time of the final **approval**

Check the requisition to determine which suppliers are involved, and choose a supplier site if there is more than one site for the specified supplier... ..each of those suppliers and use it to transmit the order.

The three ordering modules are a Purchasing Card Module, Direct Order Module, and a **Purchase Order** module.

1. Purchasing Card Module

The Purchasing Card ordering module supports the use of purchasing

cards as a payment mechanism. Purchasing cards (p-cards) can...with a supplier, then the system.

a. Checks that the transfer method has been designated for direct order in the item template. If neither the **purchase order** (PO) or DO order module has been designated in the item template then the supplier profile will be checked for the transfer method. If the... ..Department reconcile with the master statement from the supplier.

The frequency of the report will mirror the frequency of the report from the supplier.

## 2. Purchase Orders

The **purchase order** module is an ordering module whose case results in a purchase requisition in the ERP system. The system transmits the requisition to the ERP adapter... ..is in the ERP, the Purchasing Agent can manipulate it with standard ERP operations to complete the process. For example, the agent typically autocreates a **purchase order** from the requisition, prints it out, and sends it to the supplier for fulfillment.

### Receiyiqg

After an order is ...Free form comment, for noting problems. Any sort of problem will cause the item to be routed to a purchasing agent, to be handled manually.

### Approvals and Notifications

If the employee rejects an item, the system notifies Purchasing, and records the rejection.

- 42

### Reports

The reports shown in Tables 16 and...title, textual Like organization level, but textual (i.e.

Extrinsic

"Director")

10. Telephone number Phone # Extrinsic

11. Manager Is e-mail address For displaying during **approval** routing Extrinsic 12. Manager's

phone number For displaying during **approval** routing Extrinsic

ERP Adapters

ERP adapters are the pieces that integrate the system with an enterprise

ERP system. The adapters are customized for each ERP... ..requisition adapter is the basic piece that integrates with the ERP.

It pushes fully-approved requisitions into the ERP, where they are converted into - 45 **Purchase Orders** on the ERP system. The adapter can pull back the **purchase order** numbers for those requisitions, and store the PO numbers as extrinsic data fields associated with each line item.

The adapter pushes the following data for...the requester exists in the ERP. If there is no such user name in the ERP, then there will be a standard catch-all user,

## Approvals

Quantity

Unit Price

Unit of measure

Ship-to and Deliver-to addresses

Part number

Part description

Accounting information

Shipping details -- Carrier and carrier method

Supplier...mail interfaces, accounting and purchasing procedures, supplier data, client hardware and software, supported browsers, network configuration, and business rules.

b. Configure the extensible fields and **approval** rules, using a text file editor.

### 5. Seeding the database

For compatibility during the transition period, the system provides the

ability to seed the database... ..a completed paper

requisition into the system, without routing for signature. Requisitions entered in this way will appear in reports, but will not generate any **approval** requests or - 50 notifications, and will not be part of the Product Information Database without the explicit intervention of a Purchasing Agent.

6. Standalone Systems provide basic functionality when the system is stand-alone; when there is no ERP adapter present.

a. Provides the ability to print out **purchase orders** and transmit them to the supplier. The printed **purchase orders** include standard notes (such as the supplier's terms and conditions) and a **purchase order** number. This is the only time the system generates a **purchase order**.

b. Allows Purchasing Agents to modify the generated PO before it is sent to the supplier.

C. Provides a user interface for adding suppliers, providing...

## Claims:

...generating means

generating the requisition record responsive to a combination of: input by a requestor; and operating resource information in an operating resource information database; **approval** path determining means, responsive to the requisition record I and to **approval** rules in an **approval** rules database, for determining an **approval** path for the requisition record, among various ones of a plurality of possible **approvers**, required to **approve** the requisition record; **approval** path handling means for guiding the requisition record along the determined **approval** path, wherein

the **approval** path handling means generates a global **approval** indication in response to the requisition record successfully traversing the **approval** path.

2 The system of claim 1, and further comprising:

order generating means for generating an order record to a supplier of the desired operating ...  
... means for determining a method of communicating the order to the supplier, responsive to a supplier database.

5 The system of claim 1, wherein the **approval** path handling means determining means determines the **approval** path for the requisition record at least in part in response to a purchase amount field in the requisition record.

6 The system of claim... the requisition record generating means includes: means for receiving an indication of a hold time from the user via a user input means, wherein the **approval** path handling means begins to guide the requisition along the **approval** path upon occurrence of the hold time.

12 The system of claim 1, and further including:  
means for submitting a requisition in response to the global **approval** indication; wherein the requisition record generating means includes: ... the legacy database program on a periodic basis.

18 The software system of claim 14, wherein the adaptor means includes means, responsive to the global **approval** indication, for transferring the requisition to an ERP system of the enterprise.

19 The software system of claim 18, wherein the adaptor means further includes means for retrieving, from the ERP system of the enterprise, a **purchase order** number corresponding to the requisition. - 55

20 The software system of claim 18, wherein the adaptor means further includes means for retrieving supplier information from ... a supplier profile based on the supplier information retrieved from the ERP system of the enterprise.

22 The software system of claim 1, wherein:  
the **approval** rules each include a predicate and a consequence; and the **approval** path determining means determines whether a particular one of the **approval** rules applies by applying the predicate to at least one field of the requisition record; and when the **approval** path determining means determines that a particular one of the **approval** rules applies, the **approval** path determining means determines the **approval** path with respect to that **approval** rule by applying the consequence of the **approval** rule.

23 The software system of claim 1, wherein:  
the **approval** rules database includes an order definition of which, if any, required **approvers** must **approve** the requisition serially and which, if any, may **approve** the requisition in parallel; and the **approval** path handling means operates responsive to the order definition. - 56

24 The system of claim 1, wherein the **approval** path handling means includes: notification means, responsive to a position of the requisition record along the **approval** path, for notifying whichever **approver** is at that position that action is required to be taken on the requisition.

25 The software system of claim 24, wherein the **approval** path handling means includes: status change recognition means that recognizes a change in status of the requisition as a result of action taken by an **approver**, and notification means, wherein the notification means operates responsive to the status change recognition means to notify the requestor of the status change.

26 The software system of claim 1, wherein:

the action taken by the **approver** at a particular position in the **approval** path includes one of: **approving** the requisition; and denying the requisition; and the **approval** path handling means moves the requisition to a next position in the **approval** path responsive to the **approver** at the particular position **approving** the requisition.

27 The software system of claim 26, wherein:

the **approval** path handling means includes non-response handling means, responsive to an amount of time during which the requisition is at a - 57 particular position in the **approval** path without any action being taken by the **approver** at that position, for moving the requisition to another **approver** who has a predetermined relationship to the **approver** who took no action.

28 The software system of claim 27, wherein:

the predetermined relationship is indicated by chain of command data defined in an... ..29 The system of claim 26, and further comprising: notification means, wherein in response to the requisition being moved to the next position in the **approval** path, the notification means notifies the **approver** at said next position that action is required to be taken, by that **approver**, on the requisition.

30 The software system of claim 26, wherein:

the action taken by the **approver** at the particular location in the **approval** path further includes: modifying at least a portion of the requisition record; and the **approval** handling means includes modification response means, operating in response to an **approver** modifying a requisition, for causing the **approval** path determining means to determine a replacement **approval** path, responsive to the modified requisition. - 58

31 The system of claim 1, wherein the **approval** path handling means

includes non-response handling means, responsive to a request from the requestor, for moving the requisition from a first **approver** who has taken no action to a second **approver** who has a predetermined relationship to the **approver** who took no action.

32 The system of claim 31, wherein, responsive to moving the request

from the first **approver**, the **approval** path handling means prevents the first **approver** from action on the requisition.

33 The system of claim 31, wherein the predetermined relationship is at least partially defined in the **approval** rules.

34 The system of claim 1, and further including:

delegation of authority means for receiving a request from a first **approver** for delegating the authority of the first **approver** to a second **approver** by configuring the **approval** path handling means to modify the **approval** path such that the **approval** path includes the second **approver** in place of the first **approver**.

8/3K/12 (Item 9 from file: 349)

DIALOG(R)File 349: PCT FULLTEXT

(c) 2011 WIPO/Thomson. All rights reserved.

00397643

**COMPUTER-BASED SYSTEM FOR WORK PROCESSES THAT CONSIST OF  
INTERDEPENDENT DECISIONS INVOLVING ONE OR MORE PARTICIPANTS**

# SYSTEME INFORMATISE POUR PROCESSUS DE TRAVAIL CONSISTANT EN DES DECISIONS INTER-DEPENDANTES IMPLIQUANT UN OU PLUSIEURS PARTICIPANTS

## Patent Applicant/Patent Assignee:

- KONNERSMAN Paul M

## Inventor(s):

- KONNERSMAN Paul M

	Country	Number	Kind	Date
Patent	WO	9738386	A1	19971016
Application	WO	97US5969		19970410
Priorities	US	9616080		19960410

**Designated States:** (Protection type is "Patent" unless otherwise stated - for applications prior to 2004)

AU, CA, JP, US, AT, BE, CH, DE, DK, ES,  
FI, FR, GB, GR, IE, IT, LU, MC, NL, PT,  
SE

**Language** Publication Language: English

Fulltext word count: 11321

## Detailed Description:

...wood,  
steel or plastic, but products that are composed  
predominately or even entirely of data. Business plans,  
product specifications, labels, advertisements, computer  
software, consulting reports, **purchase orders**,  
quotations, requests for quotation, and publications of  
all sorts, are typical products of managerial and  
professional work processes,  
The data assemblies that are produced by...and the different capacities in which  
participants can be expected to contribute. It has proven  
useful to distinguish five decision roles, namely.

Decision Manager, Consultee, **Approver**, Inspector, and  
Informee,

The Decision manager plays the central role that  
has traditionally been associated with the term "decision  
maker," that is, making the choice ...in mind. The only requirement is that the Consultees  
feel that they have had an adequate opportunity to

influence the Decision Manager's choice.

An **Approver's** role is to prevent organizationally intolerable outcomes that might result from a decision made without the benefit of some expertise that the **Approver** has, and is not otherwise available to the Decision Manager. The other reason for an **Approver** is to assure that the decision has not been unduly influenced by the parochial interests of the Decision Manager to the detriment of the organization. The **Approver** role is like the Consultee role with two important differences. The **Approver** has veto power (i.e., he must be satisfied with the decision result and the process) and the **Approver** does not participate fully in the deliberations that take place before the decision. It is desirable for the

- 12

**Approver** to be informed about the progress and content of lengthy and complex deliberations as they go on, rather than being informed at the conclusion. However, the **Approver's** full participation in the pre-decision deliberations would severely undermine the role of the Decision Manager, since the **Approver** would essentially be taking on the role of Decision manager. (It would be better to make the Decision Manager a Consultee and make the **Approver** the Decision Manager- explicitly.)

An Informee's role is to make subsequent decisions and perform subsequent tasks in a way that is consistent with the... ..the decision moot,

The Inspector's role is to ensure that the result of a decision conforms to any published specifications.

Individuals who are called "**approvers**" are often merely inspectors. These so-called "**approvers**" are checking to see that others have done what they were supposed to have done-that is, they are checking to see that the result... ..colors. This is a different role than the one we

have outlined above in that the requirements are fully established and the so-called "**approver**" is simply

- 13

checking to see that they have been met, Unlike the **Approver's** role, these tasks could be delegated to any conscientious person. This is an ...18 is state diagram depicting the aspects of the Inspector object that change over time, FIG. 19 is state diagram depicting the aspects of the **Approver** object that change over time,



FIG, 20 is state diagram depicting the aspects of the Informee object that change over time.

-- FIG. 21 is a...string, delimited areas of a graphic).

"Decision Role' means the set of behaviors prescribed for a participant in a decision (e.g., Decision Manager, Consultee, **Approver**, Inspector, 20 Informee) . There can be any number of different roles defined for participants.

"Position' means position or job in an organization, usually designated by...the latter using the exit end, The abstract Decision Role class 121 has five concrete classes in the preferred implementation, Decision Manager 142, Consultee 143, **Approver** 144, Inspector 145, and Informee 146, These four concrete,, subclasses model the behaviors and responsibilities described in Table A. As indicated in FIGe 5, there...When

the Decision manager 142 enters the decision result, the Decision 100 object either transits 211 to Inspection 214 20 state, or transits 213 to **Approval** 215 state, or transits 212 to Standby 216 state, depending on whether the Decision 100 object has at least one Inspector 145 designated, or no Inspectors 145, but at least one **Approver** 144, or neither Inspectors 145 nor **Approvers** 25 144, respectively. From the Inspection 214 state, the Decision 100 object either transits 219 to **Approval** 215 state, or transits 220 to Standby 216 state when the result of inspection is "pass' and the Decision 100 object has, respectively, at least one **Approver** 144 or no 30 **Approvers** 144 designated. When the result of the inspection is 'fail,' the Decision i0o object in Inspection 214 state either transits 217 to Prepare Consultation 205... ...depending on whether the Decision 100 object 35 does or does not have any Consultees 143 designated, When  
- 27

a Decision 100 object is in **Approval** 215 state and ...223 to Deliberation 210 state, depending upon whether the Decision 100 object does or does not have any Consultees 143 designated. If the result in **Approval** 215 state is "grant," the Decision 100 object transits 221 to Standby 216 state. Upon completion of a Project 127, all of the Project's... ...in Standby 216 state and will transit 230 out of existence.

The states Prepare Consultation 205, Consultation

207, Deliberation 210, Inspection 214, Standby 216, and **Approval** 215 of a Decision 100 object aggregate to state InProcess 224. While a Decision 100 object is in InProcess 224 state, it may become necessary... ..110 Decision 100 is ready to enter the decision result. Upon completion of the decision entry , the Data 101 object transits 243 to Inspection or **Approval** 245 state if there is at least one Inspector 145 or one **Approver** 144 designated for the  
- 28

Decision 100. Otherwise, upon decision entry the Data 101 object in Entry Pending 241 state transits 244 to Data Release... ..result indicates "fail," the Data 101 object's state transits 248 to Historical 249 state. If all inspections result in "pass,"and there are no **Approvers** 144 designated for the Decision 100, the Data 101 object's state transits 251 to Data Release Pending 246 state. When there is at least one **Approver** 144 designated for said Decision 100, the **approval** review results are evaluated. If all required **approvals** are "granted," the Data 101 object's state transits 250 to Data Release Pending 246 state, If any **approval** is "denied", the Data 101 object's state transits 247 to Historical 249 state, When a Data 101 object's "hold/release" attribute is set...a Data 101 object from a subsequent Project 127, the former Data 101 object transits 256 to Historical 249 state, The states Inspection or **Approval** 245, Data Release Pending 246, and Standby 253 aggregate to state InProcess 257, If a Project 127 iterates across Decision 100 objects with related Data...266 state or Deliberation 270 state depending, respectively or whether the Decision 100 object does or does not have any Consultees 143 25 designated, Upon **approval** deny the Decision Manager 142 object either transits 275 or transits 276 from Standby 272 state to either Prepare Consultation 266 state or Deliberation 270...to Consultation 293 state, When end consultation occurs the Consultee 143 object transits 294 to Standby 295 state. If any related 15 inspection fails, or **approval** is denied, or the related Decision 100 object is iterated, the Consultee 143 object returns 296, 297, or 298 respectively to Dormant 291 state, The... ..aborts, the 5 Inspector 145 object transits 321 out of existence, Upon completion of the project, the Inspector 145 object transits 319 out of existence,  
**Dynamic Behavior of Approver 144 Decision Role 121**

Object. As depicted in FIG. 19 the **Approver** 144 object is instantiated 330 in Dormant 331 state, Upon decision entry it transits 332 to **Approval** 334 state, provided that there are no Inspector 145 objects related to this Decision 100 object, If there are Inspector 145 objects related to this Decision 100, the **Approver** 144 object transits 333 to **Approval** 334 state upon all inspections being passed. If all **approvals** are granted the **Approver** 144 object transits 336 to Standby 337 state. If any **approval** is denied, or the related Decision 100 object is iterated, the **Approver** 144 object returns 335 or 338 respectively to Dormant 331 state, If the related Decision 100 iterates while the **Approver** 144 object is in Standby 337 state, the **Approver** 144 object transits 339 to Dormant 311 state, The three states of the **Approver** 144 object aggregate to InProcess 341 state. If the Project 127 object of which the **Approver** 144 object is a component, aborts, the **Approver** 144 object transits 342 out of existence. Upon completion of the project, the **Approver** 144 object transits 340 out of existence.

Dynamic Behavior of Informee 146 Decision Role 121 object, Although Informees are required to act on the information ...pass it to the Data 101 object as the latter is instantiated.

A further example is the time chosen to instantiate the Inspector 145 and **Approver** 144 objects.

5 our preferred implementation instantiates them at the same time as the other Decision Role 121 objects on the expectation that there will... ..a feature that will probably be valued but need not be a part of the implementation. Similarly, the distinction made between the Inspector 145 and **Approver** 144 roles

## Claims:

...take Organization decision-making and implementation Decision responsibility for its process. Manager implementationConsultees Providing an opportunity to influence thedecision before it is Made. **Approvers** Submitting the decision for **approval**after it has been made, but before anycommitment is made to Implementation.Informees Providing timely notification of thedecision made, after it has been... ..process,veto.) Manager providing Decision Manager with relevantexpertim taking responsibility forinfluencing and accepting the result whenthe opportunity to Influence has beenprovided. **Approver** Prevent organizationally The Assuring

that the Decision Manager has intolerable outcomes that Organization not made a decision that favors parochial might result from a decision... ...organization to unacceptable risk, otherwise available to the Decision Manager, and assure Decision Making the requirements for decision that the decision has not been Manager **approval** as clear and as specific as unduly influenced by the possible, before the decision process parochial interests of the begins, and providing timely notification Decision Manager to the of **approval** or disapproval with the detriment of the organization. reasons for such disapproval. (Can veto.) Inspector Ensure that the result of the The Assuring that the... ...Consulla- Consultation lion lion Active, C Paused or Active State Suspendedm Deliberation Deliberation Entry Pending tit, Inspection Inspection '16-J5.i Inspector or **Approval** **Approval** `@i **Approval** T.A muffData Release an Ob Pending 4i: era IHistorical Table B, Object Concurrency. A method for instantiating project models as instances of a...

8/3K/13 (Item 10 from file: 349)  
 DIALOG(R) File 349: PCT FULLTEXT  
 (c) 2011 WIPO/Thomson. All rights reserved.

00344642

# **SYSTEMS AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION**

SYSTEMES ET PROCEDES DE GESTION SECURISEE DE TRANSACTIONS ET DE PROTECTION ELECTRONIQUE DES DROITS

**Patent Applicant/Patent Assignee:**

- **ELECTRONIC PUBLISHING RESOURCES INC**

**Inventor(s):**

- **GINTER Karl L**
- **SHEAR Victor H**
- **SPAHN Francis J**
- **VAN WIE David M**

	Country	Number	Kind	Date
Patent	WO	9627155	A2	<b>19960906</b>
Application	WO	96US2303		19960213
Priorities	US	95388107		19950213

**Designated States:** (Protection type is "Patent" unless otherwise stated - for applications prior to 2004)

AL, AM, AT, AU, AZ, BB, BG, BR, BY, CA,  
CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE,  
HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR,  
LS, LT, LU, LV, MD, MG, MK, MN, MW, MX,  
NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI,  
SK, TJ, TM, TR, TT, UA, UG, UZ, VN, KE,  
LS, MW, SD, SZ, UG, AZ, BY, KG, KZ, RU,  
TJ, TM, AT, BE, CH, DE, DK, ES, FR, GB,  
GR, IE, IT, LU, MC, NL, PT, SE, BF, BJ,  
CF, CG, CI, CM, GA, GN, ML, MR, NE, SN,  
TD, TG

**Language** Publication Language: English

Fulltext word count: 207972

### **Detailed Description:**

...applications.

support user interaction through: (a) "Pop-Up"  
applications which, for example, provide messages to  
users and enable users to take specific actions such  
as **approving** a transaction, N stand-alone VDE  
applications that provide administrative  
environments for user activities such as: end-user  
preference specifications for limiting the price...

8/3K/14 (Item 11 from file: 349)  
DIALOG(R)File 349: PCT FULLTEXT  
(c) 2011 WIPO/Thomson. All rights reserved.

00313636

**AUTOMATED PURCHASING CONTROL SYSTEM**  
SYSTEME DE COMMANDE AUTOMATISE POUR ACHATS

**Patent Applicant/Patent Assignee:**

- VISA INTERNATIONAL SERVICE ASSOCIATION

**Inventor(s):**

- LANGHANS Stephen

- GOODMAN Laurence M
- SHAPIRO Sigman L

	Country	Number	Kind	Date
Patent	WO	9531789	A1	<b>19951123</b>
Application	WO	95US5800		19950510
Priorities	US	94241106		19940511

**Designated States:** (Protection type is "Patent" unless otherwise stated - for applications prior to 2004)

AM, AT, AU, BB, BG, BR, BY, CA, CH, CN,  
 CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS,  
 JP, KE, KG, KP, KR, KZ, LK, LR, LT, LU,  
 LV, MD, MG, MN, MW, MX, NO, NZ, PL, PT,  
 RO, RU, SD, SE, SG, SI, SK, TJ, TM, TT,  
 UA, UG, UZ, VN, KE, MW, SD, SZ, UG, AT,  
 BE, CH, DE, DK, ES, FR, GB, GR, IE, IT,  
 LU, MC, NL, PT, SE, BF, BJ, CF, CG, CI,  
 CM, GA, GN, ML, MR, NE, SN, TD, TG

**Language** Publication Language: English

Fulltext word count: 9667

#### Detailed Description:

...of a large amount of paper.

Figs 1 is a process flow chart illustrating the steps required from start to finish for a typical corporate **purchase order**. The **approval** process in a typical company involves a number of corporate controls. A particular purchase for materials or services must be approved for that particular department...a real-time purchasing authorization and control system, The software and databases are structured to provide an automated electronic implementation of company limits and business **approval** processes, with a **dynamic**, overlapping heirarchical structure, while **approving** or disapproving purchases by employees in real-time at the time of purchase.

The entire purchasing process is re-engineered, and made a paperless process...purchasing control system, as opposed to the prior art system shown in Fig. 1, which requires either cash advances ahead of time or requisitions and **purchase orders** subsequent to the time of actual purchase. The present invention, by combining the credit card network system with a uniquely configured

database and operating software...tests are not needed for authorization, since the VIP account would be usable for all the purposes set forth in the following tests. Thus, an **approval** message 122 would be immediately generated,

The next test could be a country test 124, which may limit the cardholder to particular countries, A test...limit, the software proceeds to a test 160 which determines whether there is sufficient available credit in the company's open to buy (OTB) to **approve** the request.

If there is not a no-stop provision on the merchant category code, the next test performed is test 162 for determining if... ..limit. If the cash advance limit is not exceeded, test 164 determines whether there is sufficient open-to-buy (OTB) at the cardholder level to **approve** the request. If there is not, it is determined whether there is a cardholder over-limit for the OTB, if there is an OTB overlimit...exception report.

#### Double Custody

The bank is permitted to designate certain fields as requiring double-custody for modification, This means that two authorized CSRs must **approve** the change before it is accepted and applied.

#### Velocity Checking

Velocity checking is a risk management feature which banks use to prevent unauthorized use of... ..defined limit of transaction activity within a fixed period of time (e.g., daily), a bank may decline or refer an authorization request rather than **approving** it. A referral usually requires the merchant to call the bank to confirm the identity of the cardholder, The present invention provides a set of... ..and company may define different options for Corporate card and Purchasing card programs.

The three options are.

Refer the Authorization Request

Decline the Authorization Request

**Approve** and Report the Authorization Request

The company may also elect to specify up to 9

merchant category code group(s) (MCCG) (for a company) and...the period, (daily/monthly velocity limit counters), velocity limit for the period as defined for the

account, and authorization response code (e.g., decline, referral, **approval**).

The excessive authorization activity report lists excessive authorization activity by cardholder within the company. (The criteria for "excessive" will be based on the velocity limits...

## Claims:

...all default authorization

tests in response to a revision of said corresponding test,

7 The method of claim 1 wherein said failure

response option comprises **approving** the authorization request in said message but reporting a test failure in said step of generating reports. 8\* A system for authorizing transactions for distributing... authorization tests in response to a revision of said corresponding test.

14 The system of claim 8 wherein said failure

response option comprises means for **approving** the authorization request in said message but reporting a test failure in said means for generating reports.

15 The system of claim 8 wherein said...

? ds

Set	Items	Description
S1	9917694	PD<20021017
S2	0	AU=(CIRULLI, S OR CIRULLI S? OR (SUSAN(2N)CIRULLI)) OR BY=-(SUSAN(2N)CIRULLI)
S3	28	AU=(HAGER, D OR HAGER D? OR ((DAN OR DANNY) (2N)HAGER)) OR -BY=((DAN OR DANNY) (2N)HAGER)
S4	0	S1 AND (S2 OR S3) AND (APPROVE OR APPROVER OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING OR SIGNOFF) AND (PURCHASE(-W)(FORM OR FORMS OR ORDER OR ORDERS OR REQUISTION OR REQUISTIONS))
S5	0	S1 AND ((DAN OR DANNY) (2N)HAGER) OR (SUSAN(2N)CIRULLI)
S6	10391056	PD<20031017
S7	517	S1 AND (APPROVE OR APPROVER OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING OR SIGNOFF) AND (PURCHASE(W)(FORM OR FORMS OR ORDER OR ORDERS OR REQUISTION OR REQUISTIONS))
S8	14	S7 AND ((DYNAMIC OR DYNAMICALLY OR RECALCULATE OR RECALCULATES OR RECALCULATED OR RECALCULATING OR RECALCULATION OR RECALCULATIONS) (4N) (APPROVE OR APPROVES OR APPROVED OR APPROVER OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING))

? t s8/3,k/2,3



DIALOG(R)File 348: EUROPEAN PATENTS  
(c) 2011 European Patent Office. All rights reserved.  
8/3K/2 (Item 2 from file: 348)  
00836626

**Method and apparatus for distributing conditional work flow processes among a plurality of users**

Verfahren und Vorrichtung zum Verteilen von konditionellen Arbeitsflussprozessen zwischen mehreren Benutzern

Methode et appareil pour la distribution de processus conditionnel de flux de travail entre plusieurs utilisateurs

**Patent Assignee:**

- **Dun & Bradstreet Software Services, Inc.** (2047260)  
3445 Peachtree Street, NE; Atlanta, Georgia 30326-1276 (US)  
(applicant designated states:  
AT;BE;CH;DE;DK;ES;FI;FR;GB;GR;IE;IT;LI;LU;MC;NL;PT;SE)

**Inventor:**

- **Rossi, Charles**  
1 Indian Meadow Drive; Northborough, Massachusetts 01532; (US)
- **Vinter, Stephen**  
23 Hundred Oaks Lane; Ashland, Massachusetts 01721; (US)
- **Conte, Leonard**  
281 Green Street; Northborough, Massachusetts 01532; (US)
- **Chang, S. Jay**  
3 Duggan Road; Acton, Massachusetts 01720; (US)
- **Botzer, Robert**  
49 Delmar Avenue; Framingham, Massachusetts 01701; (US)
- **McAllister, Sandra**  
28 Lee Street; Lancaster, Massachusetts 01523; (US)
- **Dorden, Joanne**  
2 Nottingham Road; Grafton, Massachusetts 01519; (US)

**Legal Representative:**

- **Brunner, Michael John (28871)**  
GILL JENNINGS & EVERY Broadgate House 7 Eldon Street; London EC2M 7LH;  
(GB)

	Country	Number	Kind	Date
Patent	EP	774725	A2	19970521 (Basic)

	Country	Number	Kind	Date
Patent	EP	774725	A3	19981028
Application	EP	96304925		19960703
Priorities	US	557531		19951114

#### Designated States:

AT; BE; CH; DE; DK; ES; FI; FR; GB; GR;  
IE; IT; LI; LU; MC; NL; PT; SE

**International Patent Class (V7):** G06F-017/60; ;

**Abstract** ...used to determine what the next step in the workflow should be, to determine who the next step should be assigned to, to select which **approvers** on an **approval** list are used, etc. Various types of conditional comparisons may be made in order to perform this functionality. Yet another feature of the present invention...

**Abstract Word Count:** 115

**Language** Publication: English

Procedural: English

Application: English

Fulltext Availability	Available Text	Language	Update	Word Count
CLAIMS A		(English)	EPAB97	946
SPEC A		(English)	EPAB97	41563
Total Word Count (Document A) 42509				
Total Word Count (Document B) 0				
Total Word Count (All Documents) 42509				

**Specification:** ...work flow model.

An example of such a work flow is the routing of a document within an organization for the purpose of obtaining an **approval**. In a computer system for implementing a **purchase order** flow, a first user, such as a purchasing agent, initially creates a **purchase order** document. The creation of the **purchase order** document may include adding **purchase order** information to a computerized **purchase order** document image, drafting a **purchase order** computer document or a combination of both. After the **purchase order** document is created and at the command of the purchasing agent, the document may then be electronically routed to the second user, such as the purchasing manager, who could simply **approve/disapprove** the document or may add, delete or change information in the document prior to giving his/her **approval**. Finally, the document may be electronically routed to a third user, such as a finance or engineering manager, who may either **approve** or disapprove the **purchase order** document. The document may then be electronically routed back to the purchasing agent, who may then act upon the document accordingly.

A drawback associated with... ..the quantity and cost of the part. Nonetheless, both the original paper order system and its computerized implementation provide all of the information in the **purchase order** to everyone, rather than only the information relevant to each individuals specific needs. Where large documents are involved, each individual may be then forced to...used to determine what the next step in the workflow should be, to determine who the next step should be assigned to, to select which **approvers** on an **approval** list are used, etc. Various types of conditional comparisons may be made in order to perform this functionality.

Yet another feature of the present invention... activities/tasks 250 represented as messages 750 available to this user in his or her "New To Do List" To Do List window 700 are " **Approve** class registrations", "Registration confirmation," and "Select payment type for ...next activities/tasks 250 to the left of each message 750. In this example, the To Do list window 700, to the left of the "**Approve** class registrations" message 750, reveals that there are eight next activities/tasks 250 for this category, where six are new 770 and two are done... ..undertaken in this flow process. For this example, the next steps 230 are "Review Part Planning information" to be done by the manufacturing manager and "**approve** part planning" (not shown) to be done by the quality department manager.

The "review part planning information" message 750, representative of a next activity/task...alphabetical order where no sequence specific information for an activity is indicated.

For this example, the user has selected the Class Registration, Class Payment, Registration **Approval**, and Activity activities 1760 for his or her "Sample Class Registration" list 1750. Moreover, the user has chosen to sequentially list the activities such that the Class Registration Activity has the highest SEQ(underscore)NBR, with Class Payment and Registration **Approval** having lower SEQ(underscore)NBRs and Activity having the lowest or no SEQ(underscore)NBR.

From the "Sample Class Registration" list window 1750, the user...see FIG. 12). Other next steps selected by the administrator include the "Class payment type selected" (pamsam2up1) event with the next activity/task being "Registration **Approval**" (pam0520) which is assigned to user RSD.

The administrator may also activate the archiving feature of the computer system of the present invention. This feature...230 of FIG. 2), to determine to whom the next step should be assigned (e.g., element 240 of FIG. 2), or to select which **approvers** on an **approval** list should be used. In one embodiment, the following operations may be used as conditional logic:

\* Comparing an integer or numeric column to a constant... element 220 of FIG. 2) within conditional logic. An enhanced trigger event function may be implemented to allow applications to pass more than 16 values.

**Approval Lists.** Adding **approval** lists to the present invention adds the following functionality:

\* Provide **approve** and reject as activity actions. This allows the **approver** to see the object that they are **approving**. Otherwise, **approvers** in a work flow use an **approval** window and have to zoom to the actual object.

\* Provide a consistent handling of **approvals**. This ensures that all the **approval** windows have the same look and feel. It also decreases the development cost of adding **approvals** to new windows and the maintenance cost of the **approval** maintenance windows.

\* Automatic support for new **Approval** features as they are added to the platform. These include substitutes, roles, and conditional generation.

\* Allow users to **approve** items directly from the Task Details list (element 5601 of FIG. 56) without actually entering the application activity.

Structures Integration. The present invention may be...replace the previous workflow workbench with a new user interface that is more intuitive and which supports specification of conditional next steps, conditional assignments, and **approvals**.

Mail Integration. The present invention may be designed to support mail integration by creating an "attachment". Mail would be used as the delivery mechanism for...Data 10010, Workflow Definition 10015, and Message Queue tables 10050. The Workflow Engine 10045 may also include logic to resolve roles and substitutes, defined previously.

**Approvals** may be built into the workflow. **Approvals** build on the application specific **approval** processes that are already in place, and require the application developer to do the following:

- 1) Add an **approval** table in the application database. The key of **approval** table is the application table key, a sequence number, and an owner. The entries with the key matching the key of the application table row would be the **approval** list for that row.
- 2) The platform provides **approval** lists. When an **approval** is the next step the **approval** list to be used is passed to the **approval** window as data. Conditional logic can be supported in the determining of which **approval** list is to be used. **Dynamic** generation of the **approval** list will allow the name of the **approval** list to be used to be a field on the application window.
- 3) The platform will also provide a generalized window for tracking **approvals**. This window will be the management interface into the application provided **approval** table. The **approval** tracking window may either be an ancestor window that the applications use to create their own descendant tracking window with the proper keys or may...id or workgroup id of the substitute user or workgroup.

\* subst(underscore)owner(underscore)type - Indicates whether subst(underscore)owner is a user or workgroup. **approval(underscore)list**. The **approval** list table contains the header information for each **approval** list. It contains the following columns:

- \* `apprvl(underscore)list(underscore)id` - The name of the **approval** list.
- \* `apprvl(underscore)activity(underscore)id` - The activity the **approval** list is defined for.
- \* `notes` - Notes that the user can use to describe an **approval** list. **approval** (underscore)list(underscore)detail. The **approval** list detail table contains a line item for each **approver** on the list. It contains the following columns:
- \* `apprvl(underscore)list(underscore)id` - The name of the **approval** list.
- \* `apprvl(underscore)activity(underscore)id` - The activity the **approval** list is defined for.
- \* `apprvl(underscore)level` - The level for this **approver** on the list.
- \* `apprvl(underscore)id` - The user ID, workgroup or role of the **approver**.
- \* `apprvl(underscore)type(underscore)code` - The type of `apprvl(underscore)id`, either user (u), workgroup (g), role (r), or structure function (f).
- \* `structure(underscore)function...workflow engine` will have to resolve the role.

Substitute processing - If the user has a substitute defined assign change the assignee to the substitute user.

**Approvals** -- **Approvals** are described in further detail in a later section. The new algorithm for the `psp(underscore)trigger(underscore)ams(underscore)event` stored procedure is described...  
...tables and return a boolean TRUE or FALSE to indicate the value of the expression.

Conditional workflow will preferably require the following application architecture changes:

**Approve, Reject.** **Approve** and **Reject** are few actions that are only available for windows that support **approvals**. **Approve** and **Reject** will execute the logic that is currently in the **approval** windows. These actions are described in further detail in a later section.

Fix step completion bugs. The logic within basic window which controls when a... ..postfix notation before storing them in the appropriate conditional table. This work flow workbench mapping tool is defined in further detail in a later section.

**Approval Tracking.** This window may be keyed by the application key. It may display the current status of the **approval** process for the item shown. It also shows a list of the future **approvers**. **Approvals** are described in further detail in a later section.

**Approval List Definition.** This window allows the administrator to create an **approval** list. Each list is keyed by the list name and the activity class for which it is applicable. The global activity class (\*) means that the **approval** list can be used for any window in the system that supports **approvals**. **Approvals** are described in further detail in a later section.

Role Definition. The Role Definition window is a simple tabular window used to maintain the role...structure functions as the assign to in next step options requires that structures be enhanced to include work flow assignee fields within the point definition.

## **Approvals**

### Definitions

**Approval Enabled Activity** - A window that supports user configuration of **Approval** workflow by calling the new **Approval** trig(underscore)event function and allowing for the possibility that no **Approval** process was required. The activity is identified to the workflow mapping tool through the new **apprvl(underscore)enabled(underscore)activity** table.

**Approval Enabled Event** - An event that allows users to map a workflow using an **Approval** List for distribution of Next Step messages.

**Approval List** - A list of Users, Workgroups, Roles or structures functions which supports sequential grouping used in an **Approval** process for an **Approval Enabled Activity**.

**Approval Tracking/Status** - An application window/table that supports tracking of a single instance of the **Approval** process and access to **Approval** comments. Shows **Approval** level, **approver**, name and type, **Approval** Status.

**Substitutes** - A User or Workgroup that will temporarily receive Workflow messages intended for a specified User. This will apply to ALL Workflow not just **Approvals**.

**Workflow Mapping Tool** - A completely new graphical user interface for defining and describing workflows. It is intuitive and supports specification of conditional Next Steps, conditional assignments and assignment of **Approval** Lists. It is described in further detail below.

The main goal of the **Approvals** portion of the present invention is to provide the user 120 with a way to customize the **Approval** process with the least amount of work required by the applications. A secondary goal is to provide a model, with supporting tools, to standardize the way we do **Approvals**. This will allow automatic support of new **Approval** features as they are added to the platform. These include Substitutes, Roles, and conditional generation

Four application windows which use an **Approval** process have been used as the basis for the design. Currently, each transaction window that supports **approvals** has:

- 1) An **Approval** List table, stored procedures and a maintenance window with a "Copy From" response window.
- 2) An **Approval** Substitutes table, stored procedures and a maintenance window.

3) An **Approval** Tracking/Status table to monitor the **approval** process of a specific instance, a comments table, stored procedures and a maintenance window.

4) Some mechanism for either **approving** or rejecting and handling the updates to tracking/status tables.

5) Scripts to read the instance table, generate the appropriate To Do messages and to recognize when the **Approval** process is complete.

Payment Request and Journal Entry **Approvals** are very similar in that they both use the common **Approval** List/Copy From and Substitutes ancestor windows and have an **Approval** Status table with associated Comments. Purchase Requisitions use a similar format for Lists and Substitutes but have these tables in a different table family than the Status and Comments tables. They also support **Approval** at both the line and the document level. All three use the **Approval** Tracking window as the Next Step activity in the **Approval** process, thus making the object being approved only visible at **Approval** time through Zoom. Requisitions provides an **Approval** view on the Requisition maintenance window. ECR/ECN seems the most different from the others with Comments associated with the document being approved rather than the Status table, **Approval** from the document window and different column names and datatypes for Lists and Substitutes.

To provide **Approval** Enabled activities, i.e. activities which support user modification of the **Approval** process through the use of the Workflow Mapping Tool, and to integrate new workflow features in to that process, the following tasks must be accomplished:

1.) Provide a mechanism to identify **Approval** Enabled activities and events and their components, e.g. the **Approving** activity. Also, indicate when a To Do is for an **Approval** activity.

2) Consolidate multiple application **Approval** List tables into a single platform table. Add support for structures based functions that can be used within workflow to determine workflow assignees. Structures which... ..properties of the ancestor of a point at a given layer.

Determine the assignee by getting the workflow dynamic properties of a point.

3) Incorporate **Approval** List Substitutes into ...is defined, To Dos that are already in the users To Do list are not affected

4) Provide a model window for consistent handling of **Approval** Status/Tracking. This would make sure that all the **Approval** Tracking windows have the same look and feel. It would also decrease the development cost of adding **Approvals** to new windows. The existing application tracking windows can still be used. The GUI is generally consistent.

5) Provide a PowerBuilder object/ancestor to support **Approve**, Reject and Comments from within an application window which will allow a view of the object being approved. This change would allow the **approver** to see the object that they are **approving**. Currently, **approvers** use

the **Approval** Tracking window and have to Zoom to the actual object. For this release, the view of the actual object will be read-only.

6) Allow users to **Approve** items directly from the Task Details list without actually entering the application activity.

7). Provide workflow functions which will consolidate code so that an **Approval** Enabled window can use two functions to do ALL of the work required including

Generating rows for the tracking/status instance table for each **approver** on the user selected **approval** list

Generating To Dos for the first level of **approvers**

Generating any additional To Dos for Next Steps attached to the **Approval** enabled event

Returning the ID and the type of the **approver** used or an indicator that no **approval** process was required

New workflow functions may be written to support **Approvals**. Both are a variation of `pam0011(underscore)trig(underscore)event()`. The first, `pam0012(underscore)trig(underscore)event()`, will return a code indicating **Approval** Complete and will require three (3) additional arguments, `tracking(underscore)SQL(underscore)string`, `approver(underscore)used` and `approver(underscore)type(underscore)used`.

The flow of the new function will be:

- \* Do the same things that the old `trig(underscore)event` function does PLUS

- \* If the **approval** flag is set in a `next(underscore)step` then evaluate the `next(underscore)step(underscore)options` conditions to get the **Approver** ID and `approver(underscore)type(underscore)code`.

- \* If no **approval** is required, i.e. no list value is present, return 1, an indicator that the **Approval** is complete.

- \* Otherwise...

Update the `approver(underscore)used` and `approver(underscore)type(underscore)code` arguments for return to the caller .

- \* Using the application window provided string containing `svr(underscore)db(underscore)owner.stored(underscore)proc` and the formatted key values, insert rows into the Tracking table. Resolve Structures functions to provide the User/Workgroup/Role ID and the **Approval** Level for each **approver**. Set the status to N.



The second, `pam0013(underscore)notify(underscore)approvers()`, will generate the next batch of **Approval** To Dos as specified by the `apprvl(underscore)level` and will require one (1) additional argument, `next(underscore)approver (underscore)SQL(underscore)string`. It can also be used in the **Approval** window.

With the prior system, when an **approval** process is associated with an activity:

- \* From `dbupdate` event, call `update(underscore)insert(underscore)tran()` to insert a row into the database for the object to be approved.

- \* error handling

call `pam0011(underscore)trig(underscore)am(underscore)event(insertevent,... )`

select the owners from the **approval** list

build a `SQL(underscore)string` for each **approver** for the insert stored procedure for the instance table

insert a row into the instance table for each **approver**

call `pam0011(underscore)trig(underscore)am(underscore)event(sendapprmsg,... )` for each **approver** in the first batch.

- \* Using the new functions, call `update(underscore)insert(underscore)tran()` to insert a row into the database for the object to be approved

error handling

`complete(underscore)var = pam0012(underscore)trig(underscore) approval(underscore)event (insertevent ,...,SQL(underscore)string, approver(underscore)used, approver(underscore)type(underscore)code where SQL(underscore)string = am(underscore)server(underscore)db(underscore)owner(underscore)g + insert(underscore)instance(underscore)stored(underscore)procedure + & instance(underscore)keys (formatted for SQL)`

`pam0013(underscore)notify(underscore)approvers(sendmsgappr, .. , SQL(underscore)string, approval(underscore)level) where SQL(underscore)string = am(underscore)server(underscore)db(underscore)owner(underscore)g + date(underscore)instance(underscore)stored(underscore)procedure + & instance-keys (formatted... ..the values for the tracking/status table are the last parameters passed in.`

The description below provides additional guidance and notes in the implementation of **Approvals**.

**Approval Table Design**

The following tables change in the present invention to support **Approvals**:

next(underscore)step

\* The apprvl(underscore)ind column indicates whether the next(underscore)step is for an **Approval**. The mapping tool and the workflow engine handle **Approvals** slightly differently from other To Dos (see below).

message(underscore)queue(underscore)l

\* The apprvl(underscore)ind column is added to indicate that the To Do is an **Approval To Do**. The Task Detail window will allow **Approval** from there if TRUE.

The following tables are added :

apprvl(underscore)enabled(underscore)activity

The apprvl(underscore)enabled(underscore)activity table contains the IDs of Activities that are **Approval Enabled**. It contains the following columns:

\* activity(underscore)id (primary key) - The ID of an activity that has been coded to support customized **Approvals**.

\* apprvl(underscore)event(underscore)id (primary key) - The ID of an event which can generate an **Approval To Do**.

\* reapprvl(underscore)event(underscore)id - The ID of an event which will require reapproval because of changes to the object being approved.

\* apprvl(underscore)activity(underscore)id - The ID of the activity where the **Approval** should be performed, i.e. the Next Step activity.

\* task(underscore)detail(underscore)apprvl(underscore)id - The name of the PowerBuilder object that contains the application code to be used by Task Detail to support **Approvals** for this activity.

apprvl(underscore)list

The apprvl(underscore)list table contains **Approval** List IDs associated with the activities that use them. It contains the following columns:

\* apprvl(underscore)list(underscore)id (primary key) - The ID of an **Approval** List.

\* apprvl(underscore)activity(underscore)id (primary key) - The ID of an activity that has access to this list for **Approvals**.

\* notes - Text used as needed.

apprvl(underscore)list(underscore)detail

The apprvl-list(underscore)detail table contains a list of users, groups, roles or structures functions which can be associated with an **Approval** process. It contains the following columns:

- \* apprvl(underscore)list(underscore)id (primary(underscore)key) - The ID of the **Approval** List whose members are defined here.
- \* apprvl(underscore)activity(underscore)id (primary key) The activity that has access to this list.
- \* apprvl(underscore)level (primary key) - Indicates the order in which **Approvals** are performed.
- \* apprvr(underscore)id (primary key) - The user, workgroup or role ID of this **approver** or the default **approver** should a structures function fail to be resolved.
- \* apprvr(underscore)type(underscore)code (primary key) - Indicates whether **approver**(underscore)id is a user, workgroup, role or structure function.
- \* structure(underscore)function(underscore)type - Indicates the type of structure being used.
- \* structure(underscore)group... ..column ID of the data being used to resolve this structure function.
- \* layer(underscore)name - The name of the Layer.
- \* default(underscore)type - Indicates whether **approver** (underscore)id is a user, workgroup, role. Relevant when the structure function cannot be resolved and the **approver**(underscore)id is used as a default.

substitute

The Substitute tables contains a list of users who have substitutes defined for them. It contains the... ..number and datatype of the keys will vary from application to application.

- \* instance key(s) (primary key)
- \* apprvl(underscore)level (primary key)- Order in which **approvals** are performed.
- \* apprvr(underscore)id (primary key)- The user, workgroup or role ID of this **approver**.
- \* apprvr(underscore)type(underscore)code (primary key) - User, Workgroup or Role
- \* apprvl(underscore)status(underscore)code - N = None (hasnt been sent a message to review...  
...Date of the last status change
- \* actual(underscore)apprvr(underscore)id - ID of the user who performed the status update.

## End User Windows

### Activity Window **Approve/Reject** Standard

The new activity will inherit from the original activity to provide a complete, read-only view of whatever is being Approved. Functionality includes **Approve**, Reject, Comments, **Approval Complete**, generation of next level To Do messages and updates to the instance tracking table. The Yellow Sticky object is moveable so that all portions... ..seen.

#### Task Detail

plt0040(underscore)todo needs to be modified so that when the message queue row indicates that the Next Step activity is an **Approval**, an additional button will be displayed. When rows from the detail list are selected, enable the button. If **Approval** button clicked, execute the application provided code to update the **Approval** Tracking, generate trig(underscore)events and complete the **approval** process as needed. A reusable PowerBuilder object will contain the necessary application code.

#### User Substitute

The Substitute window allows a user to define a substitute... ..with the system so that an informational To Do is generated when a user is made a substitute for another user.

## Administration and Tracking Windows

### **Approval** List Definition

This window allows the administrator to create an **Approval** List. Each list is keyed by the list name and the activity ID for which it is applicable. A Popmenu and response window support Copy from one List to another.

#### Tracking/Status Ancestor

This window is keyed by the application key(s). It shows the current status of the **Approval** process for the item shown. It also shows a list of the future **approvers** and allows the current user to **Approve** or Reject if appropriate. PopMenu and Tool Bar provide Accept, Reject and Zoom to Document actions.

#### Administrator Substitute

The Administrator Substitute window is a simple... ..are assign(underscore)to type and user or workgroup.

Rule: Add, change, delete of Substitutes does NOT impact existing workflow.

#### Workflow Mapping Tool

When an **Approval** Enabled activity is mapped, the map knows which events are **Approval** Enabled. When the Insert event is chosen from the windows list of available events, the **Approval** Activity is presented as a possible next step. When chosen the windows presentation is slightly different in that **Approval** Lists are presented for the assignment of To Do messages.

#### Application Impact Issues

**Approval** Enabled activities must:

1) Have moved **Approval** Lists and Substitutes to the common Platform tables. Note: Substitutes will now apply to all workflow NOT just **Approvals**. These tables and their respective maintenance windows will replace the applications' individual implementations. This means that two windows can be dropped for each **Approval** Enabled activity.

2) Use the new TrigEvent function to manage **approval** processing and to process "**Approval** Complete" code, if the user has chosen to NOT **approve** under certain conditions. This function will return an indicator of **Approval** Complete and the ID of the **Approver** and its type code which was used to generate the Tracking Table rows. The application window will provide a string containing svr(underscore)db(underscore)... ..proc and the formatted key values to Insert rows into the Tracking table. The workflow engine will provide the User/Workgroup/Role ID and the **Approval** Level for each **approver**. Structures functions will be resolved at this time. This means that some code can be removed from an activity window. Each application designer should consider whether it would be useful to store the name of the **Approver** used.

3) Have written the PowerBuilder function which will process **Approvals** from the Task Detail window. A model will be provided and an attempt will be made to require the bare minimum of code from the... ..the-blanks with code that currently exists in other places.

Other steps which could be taken to move in the direction of an internally consistent **Approval** process are:

1) Rewrite Tracking/Status windows to use new standard.

2) Use the model for **Approve** /Reject from the document in addition to the **Approve** from the Tracking window.

#### StreamBuilder Impact Issues

##### **Approval** Activity window

Users who build activities which they wish to **Approval** Enable will need a GUI to enter and maintain the apprvl(underscore)enabled(underscore)activity table. **Approval** Enable one of the existing Sample Application windows. (psa0800(underscore)invoice). Use the new Yellow Sticky to do the **Approval** on a read-only view of the original activity using the new model. (psa0810(underscore)invoice(underscore)**approval**).

## **Approval Tracking Window**

Provide a sample of the **Approval** Tracking/Status implementation using the new Ancestor.  
(psa0850(underscore)invoice(underscore)status).

## **Task Detail Approval object**

Provide a sample of the PowerBuilder function used to **Approve** from the Task Detail window.  
(psa0800(underscore)invoice(underscore)approve).

## **Dependencies**

1) Structures integration requires that all structures that are used within workflows be replicated to all work flow servers 110 in the system.

2...

DIALOG(R)File 348: EUROPEAN PATENTS  
(c) 2011 European Patent Office. All rights reserved.  
8/3K/3 (Item 3 from file: 348)  
00377891

## **Electronic document approval system**

Elektronisches System zum Genehmigen von Dokumenten

Systeme electronique pour approuver des documents

Electronic document **approval** system

## **Patent Assignee:**

- **International Business Machines Corporation** (200120)  
Old Orchard Road; Armonk, N.Y. 10504 (US)  
(applicant designated states: DE;FR;GB;IT)

## **Inventor:**

- **Lemble, Philippe**  
Le Bel Vue B 77, avenue du Groupe Morgan; F-06700 Saint Laurent du Var; (FR)

## **Legal Representative:**

- **de Pena, Alain et al (15151)**

Compagnie IBM France Departement de Propriete Intellectuelle; F-06610 La Gaude;  
(FR)

	Country	Number	Kind	Date
Patent	EP	387462	A1	19900919 (Basic)
Patent	EP	387462	B1	19960508
Application	EP	89480045		19890314
Priorities	EP	89480045		19890314

**Designated States:**

DE; FR; GB; IT

**International Patent Class (V7):** G06F-017/21; ;

**Abstract** ...terminal connected to the system network to select a form among prestored document forms, fill said form in and then have said form mailed for **approval** by system users selected based on predefined and stored rules. The **approval** path is being permanently updated by the system. The system is made to filter access to the filled-in forms using prestored tables, and monitor the mailing and processing said filled-in forms for **approval**.

**Abstract Word Count:** 95

**Language** Publication: English

Procedural: English

Application: English

Fulltext Availability	Available Text	Language	Update	Word Count
CLAIMS A		(English)		711
SPEC A		(English)		9253
CLAIMS B		(English)	EPAB96	838
CLAIMS B		(German)	EPAB96	909
CLAIMS B		(French)	EPAB96	1048
SPEC B		(English)	EPAB96	9365
Total Word Count (Document A) 9965				
Total Word Count (Document B) 12160				
Total Word Count (All Documents) 22125				

**Specification:** ...A1

ELECTRONIC DOCUMENT **APPROVAL** SYSTEM

## Field of the Invention

This invention deals with **approval** on contents of electronically generated and mailed documents and with a system for generating, monitoring and processing said documents.

## Background of Invention

Electronic mailing systems... ..in European application No. 0,269,875 wherein shell forms are processed.

Once the form is completed, the originator has to get it approved. The **approval** process can be simple (for instance **approval** by the originator's manager), in that case, a simple electronic mail forwarding operation would do the job.

But in most instances the requirements are more complex and several levels of management or functional **approvals** may be required, e.g. by a Financial Analyst, a Budget Controller, etc... Often the **approval** process depends on data filled into the form: e.g. for a **purchase order** if the amount of a purchase requested doesn't exceed a given value, one level of management is sufficient, otherwise two levels of management are necessary, for instance. Also because of management decisions, **approval** rules for a given form can be changed without the form itself being changed.

With a conventional system, when the originator passes the document to the first **approver** (e.g. his manager) to get his signature (i.e. **approval**), he has lost control of it. He cannot be sure where the document is or who has it or if the document is hand carried or forwarded through internal mail. This prevails for any step in the **approval** process : first **approver** loses control upon transferring the document to the second **approver** and so on. Often **approvers** have to keep a paper copy of the document before passing it to the next **approver**.

In some instances **approval** has to be given by a delegate, but people may not know who the delegate is.

The **approver** list may also need to be modified during the **approval** process : an **approver** can be replaced by his manager, the order of **approval** can be changed or more information may be requested by an **approver** from another **approver** who has already signed.

Finally, all documents issued from forms and approved reach the person or the department who has to process and execute the request. First, checking must be done : data checking and **approval** process checking.

Then, if all is correct, action is taken such as keying the data into an operational system.

Generally, no information is returned to... ..satisfied, or he has to get information by phone or mail.



The document are to be kept not only during all the time the involved **approval** process is running, but also after that, during a retention period fixed for each form.

All known systems, however, lack efficient means for monitoring filled-in forms (herein referred to as documents), **dynamically** and electronically computing any **approval** path to be followed by a specific form based on its contents when filled-in, and controlling said path as well as filtering any request.... ..object of this invention is to provide a system for accessing a prestored blank form library, selecting a form, filling-in said form, computing an **approval** path based on filled-in form data and on specific predefined **approval** rules referring to user's job or function within the population of system's attached users, and monitoring and controlling the corresponding **approval** operations.

Another object is to provide a system for filtering any request for access to any filled-in form (document) based on filled-in data, stored updatable **approval** rules and requestor's identity.

In other words, this invention addresses the automation of all the steps involved in the processing of documents whose contents require complex **approvals**.

That includes documents origination, **approver** list determination, electronic signatures (i.e. **approval**) authentication, finalization operations, storage and general follow-up of the process.

More particularly the invention addresses an **approval** system for controlling the processing of a user originated document requiring signature by electronic **approval** by system selected users, in an electronic mailing system including terminals attached to a digital network, virtual machines (VM) including computer means, memory and software.... ..the population of system attached users, and means for generating processing and monitoring electronic documents to be mailed from any terminal to any user, said **approval** system including:

- means for storing and updating function tables wherein each system user's function and address are identified;
- means for storing document forms;
- means for storing predefined **approval** rules based on user function document type and document forms contents;
- terminal controllable means for selecting, accessing, filling- in, processing and mailing any selected form whose contents is to be subjected to **approval**;
- means sensitive to said mailing for addressing said function tables and, based upon said **approval** rules, for determining the **approval** path among the system attached users; and,
- means sensitive to said **approval** path determination for monitoring the mailing and processing of said filled-in form accordingly.

These and other objects, characteristics and advantages of the invention will... ..the programs;  
and

- a System-disk which contains all necessary routines for a user to run the SEALING application, i.e. routines needed by the **approval** application and further defined in this description. When a user wants to run the application, he has to make a read-only link to this...will be made apparent in the following description. But, it may already be stated that it contributes to provide a higher security level to the **approval** process. To that end, the only SQL commands the end user may use are predefined into access modules stored into the SEALDBA machine attached disks...  
...have been designed and stored in the system (SEALSYST) for further use and conversion into documents to be processed (e.g. approved) using the invention.

**Approvers** are normally designated by reference to their function, e.g. manager first, second, ... line of department No. xx. The function may be delegated, in which... ..different from the original assignee (titular) of the function.

When the system user (originator of a considered document) wishes to forward the document for proper **approval**, the **approval** path is **dynamically** computed using the contents of specific data fields within the document, and predefined **approval** rules. In the following implementation, the **approvers** may have been split into two categories : "Authorizers" and "Reviewers". Only an authorizer may accept (concur) or reject a document. A reviewer can only give an advice. A negative reviewer's opinion requires a subsequent authorizer **approval** for the document to proceed.

Finally, after being processed by the last **approver**, the document is forwarded automatically to a finalizing VM machine performing conventional operations such update, format and if required encrypt and send through the network to another network node, and, for the purpose of this invention, perform a control operation tailored to ascertain a higher security level to the **approval** system.

The above operations are summarized in Figure 4, using the facilities made available through SEALDBA, SEALSYST as well as the VM user's machine.

It should be noted therein, that the document as well as any information (e.g. functions) required for **approval** are not forwarded from one **approver** to the next. They are accessed from the general data base dynamically. This architecture enables updating constantly the **approval** path to the company's personnel (users) moves or reassignments.

From a functional standpoint the software architecture is organized as represented in Figure 5. In other words, access to the **approval** system is provided by software tools in REXX language. These tools (programs) will access the library of SEALING General Purpose Programs as well as access... ..Data System General Information GH24-5064

- SQL/Data System Terminal User's Guide SH24-5045

Represented in Figure 6 is a symbolic representation of the **approval** system centered on an SQL/DS data base accessible on Read/Write basis throughout the process. An entry for a document preparation (as will be explained later) involves read/write operations into the data base. This process step is followed by an **approval** step which can be triggered from both the document preparing user or directly from an **approver**. Once approved the document is further processed, followed-up and executed as needed. For that purpose not only the SQL/DS needs be accessed, but... ..instance VM directories other wise available. The document is finally forwarded to storage. Also available are means for updating "function" data particularly useful to the **approval** process; and consulting/analysis means to be used for instance for performing statistical operations.

Represented in Figure 7, is a mapping of the system data... ..data can be split into general tables required to run the system and specific tables which are accessed and/or generated when needed by the **approval** process. Obviously, the contents of said tables are based on preselected **approval** criteria which could be changed and therefore are not limitative. The following description is made with reference to criteria implemented in the preferred embodiment.

(A ... ..can find:

1. The LOGON tables including users identification data, (e.g. nodeid; userid).

2. Tables related to FUNCTIONS : it is herein assumed that the **approval** system users are assigned specific functions governing the **approval** rules (e.g. managerial organization). Said tables include :

- FUNCTION tables defining existing functions and providing for each function an acting person and a titular person identification.

- PREVIDEL tables registering previsions of delegations, assuming **approval** competence could be delegated from one person to another.

- HISTFUNC keeps track of any modification occurring in function tables for audit purposes (new function, delegation, change of titular).

3. Tables related to **APPROVAL** process :

APPFUTU, APPWAIT and APPDONE relate to documents in progress.

- APPFUTU contains for each document the functions to be involved (in the future) in the **approval** process.

- APPWAIT contains for each document the functions awaiting for an action on the document.

- APPDONE contains for each document the functions having already acted...It has exactly the same structure as APPDONE.

#### 4. Tables related to DOCUMENTS :

Filled-in forms provide so called documents to be subjected to the **approval** process. Each document includes at least a **HEADER** with the data defining the document. All headers are dynamically stored into **HEADERS** tables.

**COMMENTS** Tables contain for each document the comments added by the **approvers** during the **approval** process as they act on the document.

(B) In the specific Tables, one can find :

##### 1. Tables dealing with **FUNCTIONS** :

- determination of function table coded... ...contain the specific document data.

Control tables are sometimes necessary to control the data entered to prepare a document or to modify these data during **approval** process. Once defined and updated, they can be used for several different types of documents.

Follow-up tables are sometimes necessary to give follow-up... ...hereunder : (Table omitted)

An important concept of the system is the concept of "function". In a company or any group of people within which the **approval** system is needed to operate, prerogatives, e.g. **approval** competence, are assigned based on job assignments or title, herein referred to as "Function".

In an **approval** process, it is not the signature of a specific person which is required, but the signature of the person presently assigned the required function.

A... ...as a sum of prerogatives acknowledged by the company and assigned to a person. Conversely, a given person may have several different functions.

The **approval** system of this invention considers functions assigned rather than people. There are a lot of advantages to use this criterium because functions are generally more stable than people. For instance, **approval** rules defined in a company are generally related to people's level within the company (hierarchy).

In the system, a "function" is completely defined by... ...referenced by managed department number)

The system provides an interactive means to manage any other function, as soon as this function becomes necessary in an **approval** process.

Examples of **APPROVAL TABLES** of Figure 4 are represented hereunder. (see image in original document) (see image in original document) **APPHIST** : same structure as **APPDONE**.

APPFUTU : contains for each document in the **approval** process the list of functions which will have to deal with the document later on.

In this table the document is uniquely identified by the... ..Typfun) and a reference number (e.g. Service or Department number (Reffun)). An order number 1, 2, 3 ... indicates which function(s) will need to **approve** next, assuming no changes to the **approver** list has occurred. Several functions can have the same order number.

**Approver** type (Apptyp) indicates if the function is in the list as an Authorizer (A) or as a Reviewer (R). The Authorizer actions could be "Authorize" or "Reject"; while a reviewer should "**Approve**" or "Disapprove".

An index indicates if the function is mandatory or not on the **approver** list. This will be important to make changes in the **approver** list. If the function is not mandatory, any **approver** can suppress it, but if it is mandatory, either it is not possible to suppress it, or another function must replace it.

APPWAIT contains for each document in **approval** process the function(s) which are actually waiting for the document (i.e. next to act).

As APPFUTU, it contains document identification, function identification, **approver** type and mandatory indicator. It contains also the previous **approver's** name, one line personal comments from the previous **approver**, the date and time of previous action.

APPDONE : contains for each document the functions that have already acted on the document, the decisions and identification of the persons who have acted and of the titular if he is different.

As APPFUTU and APPWAIT, it contains document identification, function identification, **approver** type and mandatory indicator. It contains also the decision of the **approver** Y for Authorize or **Approve**; N for Reject or Disapprove,

the identification of the **approver**, name and employee serial number, the identification of the titular of the function, name and employee number, the delegation indicator, the action date and time.

As already mentioned, the **approval** system for the best mode of this invention is made to use predefined forms. Any user wishing to start a request for **approval** operation, will access a document form (blank)and fill-it in. A form includes predefined fields which, when filled-in will be stored into a... ..identification, i.e. type and reference of document. Each document is assigned a status (one character) which can be :

P : document is in progress in **approval** process

F : document is finalized, **approval** process is over.

S : document has been sent to an operational system

T : document has been transmitted for action (acknowledgment received from operational system).

H... ...his function (e.g. INDI for employee), his employee serial number and his name.

The table also contains the subject of the request submitted to **approval** (document) and the originating date and time.

In addition to the function tables already mentioned the system will use Specific Function Tables storing data to be used for defining the **approval** rules selected by the form user's company.

SPECIFIC FUNCTION TABLES (see image in original document) (see image in original document)

These tables allow retrieving a function reference from a parameter. They will be used during **approver** list determination.

The example shows so called Determination Tables useful to determine the manager who is responsible for a documented project.

Function CHARACTERISTIC tables enable determining parameters from the reference of a given type of function. They will be used during **approver** signature validation. In other words, these tables store predefined **approval** rules.

The example shows the amount of expenses authorized based on the considered manager's function and the company's selected rules setting expense thresholds.

From the above considerations, one may understand that any filled-in form (=document) contents is dynamically used by the system to determine the **approval** path when considered in combination with **approval** rules. In addition, the system is designed to enable adding or modifying forms, as needed, in a fairly simple way. Therefore, the various types of... ...the flow chart of Figure 9).

First, the user is recognized for being logged on a virtual machine which the system knows as a "signature" ( **approval**) machine assigned to a registered user.

The FUNCTION table is the main filter to access the documents. Said table is permanently updated to reflect the... ...used to find documents on which action has been taken already. Documents partially approved are stored in table APPDONE, while those approved by all required **approvers** (or rejected) are stored in an APPHIST or historic table.

To make the system attractive and end-user friendly from an operational standpoint, panels have...menu 1 calls PREPARE EXEC. (See Fig. 10 for a high level flow chart of the process for preparing a document to be submitted to **approval**).

This EXEC starts displaying the list of forms available to the user in a "Choose a Form Category" menu. It should be noted that the system is made to process forms of various categories like "purchasing" orders (APPR), "financial" orders (FINA), requests for getting **approval** to tailor a VM Machine to a new user (LOGO), etc... (see image in original document)

If there are too many categories to fit on... ..system checks in this case that the reference exists in the data-base and that the user is either the originator, or one of the **approvers**. Thus filtering access to said existing document.

The layout of the screens which the user is presented with, depends on the original form designer's...to the header screen

- PF3 Repeat the displayed item to create a new one
- PF4 Add a blank item
- PF5 All operations before sending in **approval** (see below)
- PF10 Display next item if it exists
- PF11 Display previous item if it exists

If the user presses PF12, the system shows the... ..image in original document)

From here, the user will be able to :

- PF1 : View the document as it can be viewed later by all the **approvers**
- PF2 : Change the document data
- PF4 : Add free comments which will be viewed to all the **approvers**
- PF5 : Prepare sending in **approval** (see below)
- PF6 : Save the data entered as a draft document (and retrieve it later)
- PF8 : Print the document. A print image of the document... ..self explanatory flow-chart of Figure 10.

Down to this point the process lead to a fully prepared document ready for being submitted to the **approval** process. Therefore, if the user presses PF5 either from the data entry panels, or from the "Process prepared document", the system performs the following operations... ..found, the system displays again the data entry panel where the field bearing an erroneous data is indicated by the cursor. Then, it determines the **approval** path based on functions involved, specific rules assigned for the type of document involved, and document data (see Figure 14).

If an error is found, the message "Unable to determine **approval** process" is displayed and no action is performed.

Finally, it determines for each function of the **approval** process, the acting person at the present time and the titular.

The result is displayed (see Figure 15) to the considered user (originator) as shown...  
...document)

If the user presses PF12, he just returns to "Process the document" Menu.

Otherwise, by depressing PF1 he will be able to change the **approver** list with some controls and restrictions (see below with reference to Figures 15 and 16).

If he does press PF7, the document is created and waits for action of the first function(s) in the **approval** process (see Figures 15 and 16).

Represented in Figure 11 is a flow-chart summarizing the operations achievable on an already filled-in document. The... ..functional key labeled "Open the Mail", assuming SEALING documents are waiting action. The message looks like:

"You have 5 documents awaiting your action in Electronic **Approval** System. Do you wish to process these ? Type Y and press ENTER if you wish". The system then shows first a list of categories of... ..document)

The user can see on this screen the document type and reference, the subject, the originator's name, who was the last previous **approver** and personal comments from this previous **approver**. Also mentioned is the function the present user is asked to act for, and eventually the owner for the function if the user is a delegate.

By depressing PF1, the user can view all information about the document, i.e. the document itself; the **approver** list with decisions of **approvers** who have already acted on the document; the document originator and **approvers** comments, if any.

Depressing PF4, enables the user adding his own comments to the other **approver's** comments after acting on the document.

PF5 is not available for all **approvers**. This PF Key is only active if the document is originally tailored to authorize the function to add or modify data. In this case, the user is shown data modification panels and can add or modify data in the document. These modifications can affect the **approval** path process.

PF7 must be used to act on the document.

If the user is an Authorizer, he will see the screen below. (see image in original document)



The user can change the **approvers** list by pressing PF7 (seen later).

The user can Authorize (**Approve**) the document with PF1. The system will display a confirmation panel which looks as follows. (see image in original document)

The next screen shows the next **approver** and allows the user to type one line of personal comments for said next **approver's** attention.

If the user is the last **approver**, the confirmation panel is different and looks as follows : (see image in original document)

Authorization means finalization of the document.

An Authorizer can also Reject... ..his decision, indicating that in case of non confirmation, the document will be rejected, an information will be sent to the originator and to each **approver** who has already acted on the document.

The user may also choose PF3 to request additional information from another **approver**.

In this case, the user has to choose an **approver** in the screen below. (see image in original document)

He can choose the originator, an **approver** who has already acted on the document or an **approver** who has not yet seen the document, then he gets a confirmation panel as below. (see image in original document)

If the user confirms, the document is available for the chosen **approver** and the user will get it back again after this **approver** has acted upon said document. If the user is a Reviewer in the **approval** process, he is shown the following screen. (see image in original document)

There will be two different confirmation panels no matter whether the user is the last **approver** or not. Request for additional information operation is quite the same as requested to an Authorizer. It should be remembered that a Reviewer cannot reject a document. He can just disapprove the document, in which case a further Authorizer **approval** is required. The system determines if there is an authorizer in the list of next **approvers**. If there is no Authorizer in the list of next **approvers**, the system asks the user to choose among the Authorizers who have already acted on the document and will send the document back to the... ..Or the document has been rejected by an Authorizer or has been cancelled by the originator and the user is either the Originator or an **Approver** who has already acted on it.

PF4 has just the effect to cancel reference reminder to said documents. The user can always access the document... ..can access documents using several alternatives (see Figure 12):

- PF1 : Lets the user access the documents he has originated and which are currently in the **approval** process.

NB: by user one means the person assigned corresponding function.

- PF2 : Lets the user access the documents he has processed and which are currently in the **approval** process.

- PF3 : Lets the user access all the documents, whatever their status (in progress, finalized, rejected, cancelled...) he has originated between two dates the user...the status is "In progress", the user is presented with the screen below. (see image in original document)

The user can

- view the document (data, **approvers** list and comments).
- copy the document to create a draft
- print the document

If he is the originator, he can also cancel the document. In... ..user is presented the following confirmation panel. (see image in original document)

Pressing PF7 in "Process the document found" menu allows the originator or an **approver** who has already acted on the document to change the **approver** list as he could have done when originating the document or acting upon the document.

When changes are prepared, the system displays the following panels. (see image in original document)

If all changes are correct, the user can press PF1 and the document proceeds with the new **approver** list.

PF12 will cancel all the changes.

If the status of the document is not "In progress", the available actions are different as shown hereafter. (see image in original document)

The user can

- view the document (data, **approver** list and comments).
- copy the document to create a draft
- print the document

Follow-up information may also be given when the user presses PF4... ..looks for documents awaiting action and give the references. For obvious security purposes, no further information about document can be obtained.

As already mentioned an **approver** designation, could be delegated from one user to another under predefined conditions. The user can access this part of the system by pressing PF7... ..In this case, the system considers the user as absent. This feature will be used to bar access to the document for instance when an **approver** is acting on it. So, originators and **approvers** will be informed and will be able to react to this situation.

To validate his entries, the user has to press the enter key.

Using... ..the transfer.

Having thus described from a technical as well as functional standpoint, the means involved in this invention, one may therefore fully comprehend the **approval** system operation. **Approval** system operations has however been summarized in Figures 14 through 17, and Function management in Figure 18.

Represented in Figures 14 through 16 are, the means involved in the determination of the list of **approvers**. Obviously, the system has been limited to simple cases to simplify explaining the operation. One assumes the filled-in document (e.g. **Purchase Order**) includes a Project code, a Purchaser code and an Amount of expenses set by the purchase requesting user. The software means uses the ... FaMANAD2) needs be addressed. Also, due to the type of document involving expenses, the logic adds an Investment Responsible (INVT) to the list of required **approvers**. Same applies to Purchaser (PURC). The system loads the information into two separate tables located in the document originator VM user's machine memory designated as FUTU and DONE respectively. The DONE table contains so called "shadow" **approvers** or virtual **approvers** whom the system will enable read-only access to the corresponding document. As already mentioned, once initiating the **approval** process, the originator may amend the list of **approvers** up to a certain extent based on predefined rules. For instance, deletion of a first line manager from the list of **approvers** will trigger automatic insertion of the second line manager, and so on.

Then, the originator sending decision has the effect to unload the first **approver** references from FUTU table into a NEXTWAIT table while the others are loaded into a NEXTFUTU table in the preset ordered list, both tables NEXT... ..and updating the corresponding SQL data base tables, i.e. APPFUTU, APPWAIT and APPDONE of the SEALBDA machine.

The SEALING system also controls mailing **approval** requests to designated **approvers** whose action are controlled as represented in Figure 16.

The designated **approver** in NEXTWAIT gets access to the data in the corresponding SQL/DS tables in its own VM machines into FUTU, WAIT and DONE tables. Then NEXTFUTU, NEXTWAIT and NEXT DONE are set from FUTU, WAIT and DONE tables respectively. These tables contents are used by current **approver** to perform and record the following operations:

- **Approver** list modification: the **approver** may amend the list as the originator was able to do.

- Modification to **approval** process: the system controlled by any amendment to the data of document due to current **approver**, amends the **approval** path accordingly, and

- **Approver** validation: compliance test with the **approver's** designation rules is performed.

Tables updating are performed, i.e. once current **approval** is executed, a shift is operated with next **approver** in NEXTFUTU being shifted into NEXTWAIT.

Once forwarded, the current **approval** data are used to update the SQL/DS tables through add, modify or delete operations. They are then available for next **approver's** action and so on.

A fairly high security level has been provided in this invention to limit false **approvals** due either to human error or to voluntary operation. First, one should notice that **approval**, i.e. insertion of a "Y" or "N" flag into a predetermined field characterizing the considered document is inserted into the SQL/DS Table APPDONE otherwise accessible to the user on a Read-Only basis (see MYSIGNAT in Figure 6). Second, said "signature" or **approval** insertion is operated only upon execution of a predetermined SQL command involving a signature validity check. The flow chart of such a signature validation operation... ..The SQL command triggered by the MYSIGNAT order accesses various SQL tables to gather data stored therein to ensure that the user presently trying to **approve** or disapprove is entitled to do so. First APPWAIT table is accessed to ensure that the considered document defined by typdoc/refdoc is waiting therein... ..Table, together with data and time are enabled upon a positive signature validation test result.

As mentioned, the population of users attached to the same **approval** system is managed in an unique SQL table, i.e. the FUNCTION Table, as represented in Figure 18. For each function, i.e. Manager, Purchaser...

### **Specification: ...B1**

#### **Field of the Invention**

This invention deals with **approval** on contents of electronically generated and mailed documents and with a system for generating, monitoring and processing said documents.

#### **Background of Invention**

Electronic mailing systems... ..in European application No. 0,269,875 wherein shell forms are processed.

Once the form is completed, the originator has to get it approved. The **approval** process can be simple (for instance **approval** by the originator's manager), in that case, a simple electronic mail forwarding operation would do the job.

But in most instances the requirements are more complex and several levels of management or functional **approvals** may be required, e.g. by a Financial Analyst, a Budget Controller, etc... ..Often the **approval** process depends on data filled into the form: e.g. for a **purchase order** if the

amount of a purchase requested doesn't exceed a given value, one level of management is sufficient, otherwise two levels of management are necessary, for instance. Also because of management decisions, **approval** rules for a given form can be changed without the form itself being changed.

With a conventional system, when the originator passes the document to the first **approver** (e.g. his manager) to get his signature (i.e. **approval**), he has lost control of it. He cannot be sure where the document is or who has it or if the document is hand carried or forwarded through internal mail. This prevails for any step in the **approval** process : first **approver** loses control upon transferring the document to the second **approver** and so on. Often **approvers** have to keep a paper copy of the document before passing it to the next **approver**.

In some instances **approval** has to be given by a delegate, but people may not know who the delegate is.

The **approver** list may also need to be modified during the **approval** process : an **approver** can be replaced by his manager, the order of **approval** can be changed or more information may be requested by an **approver** from another **approver** who has already signed.

Finally, all documents issued from forms and approved reach the person or the department who has to process and execute the request. First, checking must be done : data checking and **approval** process checking.

Then, if all is correct, action is taken such as keying the data into an operational system.

Generally, no information is returned to... ..satisfied, or he has to get information by phone or mail.

The document are to be kept not only during all the time the involved **approval** process is running, but also after that, during a retention period fixed for each form.

Disclosed in the Patent US-A-4,503,499 is... ..disclosed in Patent Abstracts of Japan, Vol. 9, No 206(P/149), of October 19, 1982, is a system wherein the document meant to receive **approval** by circulation is inputted together with circulation order facilities.

All known systems, however, lack efficient means for monitoring filled-in forms (herein referred to as documents), then computing fully dynamically and electronically the specific and correct **approval** path to be followed by each considered form based on its contents when filled-in, with reference to predefined sets of **approval** rules to be combined with varying titles or hierarchical positions of respective users to be automatically selected within the system users population, and controlling said... ..object of this invention is to provide a system for accessing a prestored blank form library, selecting a form, filling-in said form, computing an **approval** path based on filled-in form data and on specific predefined **approval** rules referring to user's job or function within the population of system's attached users, and monitoring and controlling the corresponding **approval** operations.

Another object of the invention is to provide a system for filtering any request for access to any filled-in form (document) based on filled-in data, stored updatable **approval** rules and requestor's identity.

In other words, this invention addresses the automation of all the steps involved in the processing of documents whose contents require complex **approvals**.

That includes documents origination, **approver** list determination, electronic signatures (i.e. **approval**) authentication, finalization operations, storage and general follow-up of the process.

More particularly the invention, as claimed, addresses an **approval** system for controlling the processing of a user originated document requiring electronic **approval** by system selected ...the population of system attached users, and means for generating processing and monitoring electronic documents to be mailed from any terminal to any user, said **approval** system including:

- means for storing and updating function tables wherein each system user's function and address are identified;
- means for storing document forms;
- means for storing predefined **approval** rules based on user function document type and document forms contents;
- terminal controllable means for selecting, accessing, filling-in, processing and mailing any selected form whose contents is to be subjected to **approval**;
- means sensitive to said mailing for addressing said function tables and, based upon said **approval** rules, for determining the **approval** path among the system attached users; and,
- means sensitive to said **approval** path determination for monitoring the mailing and processing of said filled-in form accordingly.

These and other objects, characteristics and advantages of the invention will...the programs; and

- a System-disk which contains all necessary routines for a user to run the SEALING application, i.e. routines needed by the **approval** application and further defined in this description. When a user wants to run the application, he has to make a read-only link to this... will be made apparent in the following description. But, it may already be stated that it contributes to provide a higher security level to the **approval** process. To that end, the only SQL commands the end user may use are predefined into access modules stored into the SEALDBA machine attached disks... have been designed and stored in the system (SEALSYST) for further use and conversion into documents to be processed (e.g. approved) using the invention.

**Approvers** are normally designated by reference to their function, e.g. manager first, second, ... line of department No. xx. The function may be delegated, in which... ..different from the original assignee (titular) of the function.

When the system user (originator of a considered document) wishes to forward the document for proper **approval**, the **approval** path is **dynamically** computed using the contents of specific data fields within the document, and predefined **approval** rules. In the following implementation, the **approvers** may have been split into two categories : "Authorizers" and "Reviewers". Only an authorizer may accept (concur) or reject a document. A reviewer can only give an advice. A negative reviewer's opinion requires a subsequent authorizer **approval** for the document to proceed.

Finally, after being processed by the last **approver**, the document is forwarded automatically to a finalizing VM machine performing conventional operations such update, format and if required encrypt and send through the network to another network node, and, for the purpose of this invention, perform a control operation tailored to ascertain a higher security level to the **approval** system.

The above operations are summarized in Figure 4, using the facilities made available through SEALDBA, SEALSYST as well as the VM user's machine.

It should be noted therein, that the document as well as any information (e.g. functions) required for **approval** are not forwarded from one **approver** to the next. They are accessed from the general data base dynamically. This architecture enables updating constantly the **approval** path to the company's personnel (users) moves or reassignments.

From a functional standpoint the software architecture is organized as represented in Figure 5. In other words, access to the **approval** system is provided by software tools in "Restructured Extended Executor (RSXX)" language. These tools (programs) will access the library of SEALING General Purpose Programs as... ..Data System General Information GH24-5064

- SQL/Data System Terminal User's Guide SH24-5045

Represented in Figure 6 is a symbolic representation of the **approval** system centered on an SQL/DS data base accessible on Read/Write basis throughout the process. An entry for a document preparation (as will be explained later) involves read/write operations into the data base. This process step is followed by an **approval** step which can be triggered from both the document preparing user or directly from an **approver**. Once approved the document is further processed, followed-up and executed as needed. For that purpose not only the SQL/DS needs be accessed, but... ..for instance VM directories otherwise available. The document is finally forwarded to storage. Also available are means for updating "function" data particularly useful to the **approval** process; and consulting/analysis means to be used for instance for performing statistical operations.

Represented in Figure 7, is a mapping of the system data... ..data can be split into general tables required to run the system and specific tables which are accessed and/or generated when needed

by the **approval** process. Obviously, the contents of said tables are based on preselected **approval** criteria which could be changed and therefore are not limitative. The following description is made with reference to criteria implemented in the preferred embodiment.

(A... ..can find:

1. The LOGON tables including users identification data, (e.g. nodeid; userid).

2. Tables related to FUNCTIONS : it is herein assumed that the **approval** system users are assigned specific functions governing the **approval** rules (e.g. managerial organization). Said tables include :

FUNCTION tables defining existing functions and providing for each function an acting person and a titular person identification.

PREVIDEL tables registering previsions of delegations, assuming **approval** competence could be delegated from one person to another.

HISTFUNC keeps track of any modification occurring in function tables for audit purposes (new function, delegation, change of titular).

3. Tables related to **APPROVAL** process :

APPFUTU, APPWAIT and APPDONE relate to documents in progress.

APPFUTU contains for each document the functions to be involved (in the future) in the **approval** process.

APPWAIT contains for each document the functions awaiting for an action on the document.

APPDONE contains for each document the functions having already acted... ..It has exactly the same structure as APPDONE.

4. Tables related to DOCUMENTS :

Filled-in forms provide so called documents to be subjected to the **approval** process. Each document includes at least a HEADER with the data defining the document. All headers are dynamically stored into HEADERS tables.

COMMENTS Tables contain for each document the comments added by the **approvers** during the **approval** process as they act on the document.

(B) In the specific Tables, one can find :

1. Tables dealing with FUNCTIONS :



determination of function table coded... ..contain the specific document data.

Control tables are sometimes necessary to control the data entered to prepare a document or to modify these data during **approval** process. Once defined and updated, they can be used for several different types of documents.

Follow-up tables are sometimes necessary to give follow-up... ..in original document)

An important concept of the system is the concept of "function". In a company or any group of people within which the **approval** system is needed to operate, prerogatives, e.g. **approval** competence, are assigned based on job assignments or title, herein referred to as "Function".

In an **approval** process, it is not the signature of a specific person which is required, but the signature of the person presently assigned the required function.

A... ..defined as a sum of prerogatives acknowledged by the company and assigned to a person. Conversely, a given person may have several different functions.

The **approval** system of this invention considers functions assigned rather than people. There are a lot of advantages to use this criterium because functions are generally more stable than people. For instance, **approval** rules defined in a company are generally related to people's level within the company (hierarchy).

In the system, a "function" is completely defined by... ..referenced by managed department number)

The system provides an interactive means to manage any other function, as soon as this function becomes necessary in an **approval** process.

Examples of **APPROVAL TABLES** of Figure 4 are represented hereunder.

## **APPROVAL TABLES**

(see image in original document) (see image in original document)

APPHIST : same structure as APPDONE.

APPFUTU : contains for each document in the **approval** process the list of functions which will have to deal with the document later on.

In this table the document is uniquely identified by the... Typfun) and a reference number (e.g. Service or Department number (Reffun)). An order number 1, 2, 3 ... indicates which function(s) will need to **approve** next, assuming no changes to the **approver** list has occurred. Several functions can have the same order number.

**Approver** type (Apptyp) indicates if the function is in the list as an Authorizer (A) or as a Reviewer (R). The Authorizer actions could be "Authorize" or "Reject"; while a reviewer should "**Approve**" or "Disapprove".

An index indicates if the function is mandatory or not on the **approver** list. This will be important to make changes in the **approver** list. If the function is not mandatory, any **approver** can suppress it, but if it is mandatory, either it is not possible to suppress it, or another function must replace it.

APPWAIT contains for each document in **approval** process the function(s) which are actually waiting for the document (i.e. next to act).

As APPFUTU, it contains document identification, function identification, **approver** type and mandatory indicator. It contains also the previous **approver's** name, one line personal comments from the previous **approver**, the date and time of previous action.

APPDONE : contains for each document the functions that have already acted on the document, the decisions and identification of the persons who have acted and of the titular if he is different.

As APPFUTU and APPWAIT, it contains document identification, function identification, **approver** type and mandatory indicator. It contains also the decision of the **approver** Y for Authorize or **Approve**; N for Reject or Disapprove,

the identification of the **approver**, name and employee serial number, the identification of the titular of the function, name and employee number, the delegation indicator, the action date and time.

As already mentioned, the **approval** system for the best mode of this invention is made to use predesigned forms. Any user wishing to start a request for **approval** operation, will access a document form (blank) and fill-it in. A form includes predefined fields which, when filled-in will be stored into a ... ..identification, i.e. type and reference of document. Each document is assigned a status (one character) which can be :

P : document is in progress in **approval** process

F : document is finalized, **approval** process is over.

S : document has been sent to an operational system

T : document has been transmitted for action (acknowledgment received from operational system).

H... ..his function (e.g. INDI for employee), his employee serial number and his name.

The table also contains the subject of the request submitted to **approval** (document) and the originating date and time.

In addition to the function tables already mentioned the system will use Specific Function Tables storing data to be used for defining the **approval** rules selected by the form user's company. (see image in original document)

These tables allow retrieving a function reference from a parameter. They will be used during **approver** list determination.

The example shows so called Determination Tables useful to determine the manager who is responsible for a documented project.

Function CHARACTERISTIC tables enable determining parameters from the reference of a given type of function. They will be used during **approver** signature validation. In other words, these tables store predefined **approval** rules.

The example shows the amount of expenses authorized based on the considered manager's function and the company's selected rules setting expense thresholds.

From the above considerations, one may understand that any filled-in form (=document) contents is dynamically used by the system to determine the **approval** path when considered in combination with **approval** rules. In addition, the system is designed to enable adding or modifying forms, as needed, in a fairly simple way. Therefore, the various types of... ..the flow chart of Figure 9).

First, the user is recognized for being logged on a virtual machine which the system knows as a "signature" ( **approval**) machine assigned to a registered user.

The FUNCTION table is the main filter to access the documents. Said table is permanently updated to reflect the...used to find documents on which action has been taken already. Documents partially approved are stored in table APPDONE, while those approved by all required **approvers** (or rejected) are stored in an APPHIST or historic table.

To make the system attractive and end-user friendly from an operational standpoint, panels have... ..menu 1 calls PREPARE EXEC. (See Fig. 10 for a high level flow chart of the process for preparing a document to be submitted to **approval**).

This EXEC starts displaying the list of forms available to the user in a "Choose a Form Category" menu. It should be noted that the system is made to process forms of various categories like "purchasing" orders (APPR), "financial" orders (FINA), requests for getting **approval** to tailor a VM Machine to a new user (LOGO), etc... (see image in original document)

If there are too many categories to fit on... ..system checks in this case that the reference exists in the data-base and that the user is either the originator, or one of the **approvers**. Thus filtering access to said existing document.

The layout of the screens which the user is presented with, depends on the original form designer's...to the header screen

- PF3 Repeat the displayed item to create a new one
- PF4 Add a blank item
- PF5 All operations before sending in **approval** (see below)
- PF10 Display next item if it exists
- PF11 Display previous item if it exists

If the user presses PF12, the system shows the... ..image in original document)

From here, the user will be able to :

- PF1 : View the document as it can be viewed later by all the **approvers**
- PF2 : Change the document data
- PF4 : Add free comments which will be viewed to all the **approvers**
- PF5 : Prepare sending in **approval** (see below)
- PF6 : Save the data entered as a draft document (and retrieve it later)
- PF8 : Print the document. A print image of the document... ..self explanatory flow-chart of Figure 10.

Down to this point the process lead to a fully prepared document ready for being submitted to the **approval** process. Therefore, if the user presses PF5 either from the data entry panels, or from the "Process prepared document", the system performs the following operations... ..found, the system displays again the data entry panel where the field bearing an erroneous data is indicated by the cursor. Then, it determines the **approval** path based on functions involved, specific rules assigned for the type of document involved, and document data (see Figure 14).

If an error is found, the message "Unable to determine **approval** process" is displayed and no action is performed.

Finally, it determines for each function of the **approval** process, the acting person at the present time and the titular.

The result is displayed (see Figure 15) to the considered user (originator) as shown... ..document)

If the user presses PF12, he just returns to "Process the document" Menu.

Otherwise, by depressing PF1 he will be able to change the **approver** list with some controls and restrictions (see below with reference to Figures 15 and 16).

If he does press PF7, the document is created and waits for action of the first function(s) in the **approval** process (see Figures 15 and 16).

Represented in Figure 11 is a flow-chart summarizing the operations achievable on an already filled-in document. The... ..functional key labeled "Open the Mail", assuming SEALING documents are waiting action. The message looks like:

"You have 5 documents awaiting your action in Electronic **Approval** System. Do you wish to process these ? Type Y and press ENTER if you wish". The system then shows first a list of categories of... ..document)

The user can see on this screen the document type and reference, the subject, the originator's name, who was the last previous **approver** and personal comments from this previous **approver**. Also mentioned is the function ...function if the user is a delegate.

By depressing PF1, the user can view all information about the document, i.e. the document itself; the **approver** list with decisions of **approvers** who have already acted on the document; the document originator and **approvers** comments, if any.

Depressing PF4, enables the user adding his own comments to the other **approver's** comments after acting on the document.

PF5 is not available for all **approvers**. This PF Key is only active if the document is originally tailored to authorize the function to add or modify data. In this case, the user is shown data modification panels and can add or modify data in the document. These modifications can affect the **approval** path process.

PF7 must be used to act on the document.

If the user is an Authorizer, he will see the screen below. (see image in original document)

The user can change the **approvers** list by pressing PF7 (seen later).

The user can Authorize (**Approve**) the document with PF1. The system will display a confirmation panel which looks as follows. (see image in original document)

The next screen shows the next **approver** and allows the user to type one line of personal comments for said next **approver's** attention.

If the user is the last **approver**, the confirmation panel is different and looks as follows : (see image in original document)

Authorization means finalization of the document.

An Authorizer can also Reject... ..his decision, indicating that in case of non confirmation, the document will be rejected, an information will be sent to the originator and to each **approver** who has already acted on the document.

The user may also choose PF3 to request additional information from another **approver**.

In this case, the user has to choose an **approver** in the screen below. (see image in original document)

He can choose the originator, an **approver** who has already acted on the document or an **approver** who has not yet seen the document, then he gets a confirmation panel as below. (see image in original document)

If the user confirms, the document is available for the chosen **approver** and the user will get it back again after this **approver** has acted upon said document. If the user is a Reviewer in the **approval** process, he is shown the following screen. (see image in original document)

There will be two different confirmation panels no matter whether the user is the last **approver** or not. Request for additional information operation is quite the same as requested to an Authorizer. It should be remembered that a Reviewer cannot reject a document. He can just disapprove the document, in which case a further Authorizer **approval** is required. The system determines if there is an authorizer in the list of next **approvers**. If there is no Authorizer in the list of next **approvers**, the system asks the user to choose among the Authorizers who have already acted on the document and will send the document back to the... ..Or the document has been rejected by an Authorizer or has been cancelled by the originator and the user is either the Originator or an **Approver** who has already acted on it.

PF4 has just the effect to cancel reference reminder to said documents. The user can always access the document... ..can access documents using several alternatives (see Figure 12):

- PF1 : Lets the user access the documents he has originated and which are currently in the **approval** process.

NB: by user one means the person assigned corresponding function.

- PF2 : Lets the user access the documents he has processed and which are currently in the **approval** process.

- PF3 : Lets the user access all the documents, whatever their status (in progress, finalized, rejected, cancelled...) he has originated between two dates the user ...the status is "In progress", the user is presented with the screen below. (see image in original document)

The user can - view the document (data, **approvers** list and comments).

- copy the document to create a draft

- print the document

If he is the originator, he can also cancel the document. In... user is presented the following confirmation panel. (see image in original document)

Pressing PF7 in "Process the document found" menu allows the originator or an **approver** who has already acted on the document to change the **approver** list as he could have done when originating the document or acting upon the document.

When changes are prepared, the system displays the following panels. (see image in original document)

If all changes are correct, the user can press PF1 and the document proceeds with the new **approver** list.

PF12 will cancel all the changes.

If the status of the document is not "In progress", the available actions are different as shown hereafter. (see image in original document)

The user can - view the document (data, **approver** list and comments).

- copy the document to create a draft

- print the document

Follow-up information may also be given when the user presses PF4... looks for documents awaiting action and give the references. For obvious security purposes, no further information about document can be obtained.

As already mentioned an **approver** designation, could be delegated from one user to another under predefined conditions. The user can access this part of the system by pressing PF7 on... In this case, the system considers the user as absent. This feature will be used to bar access to the document for instance when an **approver** is acting on it. So, originators and **approvers** will be informed and will be able to react to this situation.

To validate his entries, the user has to press the enter key.

Using... the transfer.

Having thus described from a technical as well as functional standpoint, the means involved in this invention, one may therefore fully comprehend the **approval** system operation. **Approval** system operations has however been summarized in Figures 14 through 17, and Function management in Figure 18.

Represented in Figures 14 through 16 are, the means involved in the determination of the list of **approvers**. Obviously, the system has been limited to simple cases to simplify explaining the operation. One assumes the filled-in document (e.g. **Purchase Order**) includes a Project code, a Purchaser code and an Amount of expenses set by the purchase requesting user. The software means uses the project code... ..FaMANAD2) needs be addressed. Also, due to the type of document involving expenses, the logic adds an Investment Responsible (INVT) to the list of required **approvers**. Same applies to Purchaser (PURC). The system loads the information into two separate tables located in the document originator VM user's machine memory designated as FUTU and DONE respectively. The DONE table contains so called "shadow" **approvers** or virtual **approvers** whom the system will enable read-only access to the corresponding document. As already mentioned, once initiating the **approval** process, the originator may amend the list of **approvers** up to a certain extent based on predefined rules. For instance, deletion of a first line manager from the list of **approvers** will trigger automatic insertion of the second line manager, and so on.

Then, the originator sending decision has the effect to unload the first **approver** references from FUTU table into a NEXTWAIT table while the others are loaded into a NEXTFUTU table in the preset ordered list, both tables NEXT... ..building and updating the corresponding SQL data base tables, i.e. APPFUTU, APPWAIT and APPDONE of the SEALBDA machine.

The SEALING system also controls mailing **approval** requests to designated **approvers** whose action are controlled as represented in Figure 16.

The designated **approver** in NEXTWAIT gets access to the data in the corresponding SQL/DS tables in its own VM machines into FUTU, WAIT and DONE tables. Then NEXTFUTU, NEXTWAIT and NEXT DONE are set from FUTU, WAIT and DONE tables respectively. These tables contents are used by current **approver** to perform and record the following operations:

- **Approver** list modification: the **approver** may amend the list as the originator was able to do.
- Modification to **approval** process: the system controlled by any amendment to the data of document due to current **approver**, amends the **approval** path accordingly, and
- **Approver** validation: compliance test with the **approver**'s designation rules is performed.

Tables updating are performed, i.e. once current **approval** is executed, a shift is operated with next **approver** in NEXTFUTU being shifted into NEXTWAIT.

Once forwarded, the current **approval** data are used to update the SQL/DS tables through add, modify or delete operations. They are then available for next **approver**'s action and so on.

A fairly high security level has been provided in this invention to limit false **approvals** due either to human error or to voluntary operation. First, one should notice that **approval**, i.e. insertion of a "Y" or "N" flag into a predetermined field characterizing the considered document is inserted into the SQL/DS Table APPDONE otherwise accessible to the user on a Read-Only



basis (see MYSIGNAT in Figure 6). Second, said "signature" or **approval** insertion is operated only upon execution of a predetermined SQL command involving a signature validity check. The flow chart of such a signature validation operation... ..The SQL command triggered by the MYSIGNAT order accesses various SQL tables to gather data stored therein to ensure that the user presently trying to **approve** or disapprove is entitled to do so. First APPWAIT table is accessed to ensure that the considered document defined by typdoc/refdoc is waiting therein... ..Table, together with data and time are enabled upon a positive signature validation test result.

As mentioned, the population of users attached to the same **approval** system is managed in an unique SQL table, i.e. the FUNCTION Table, as represented in Figure 18. For each function, i.e. Manager, Purchaser...

#### Claims: ...A1

1. An **approval** system for controlling the processing of a user originated document requiring signature by electronic **approval** by system selected users, in an electronic mailing system including terminals attached to a digital network, virtual machines (VM) including computer means, memory and software... ..the population of system attached users, and means for generating processing and monitoring electronic documents to be mailed from any terminal to any user, said **approval** system including:

- means for storing and updating function tables wherein each system user's function and address are identified;
- means for storing document forms;
- means for storing predefined **approval** rules based on user function and document forms contents;
- terminal controllable means for selecting, accessing, filling-in, processing and mailing any selected form whose contents is to be subjected to **approval**;
- means sensitive to said mailing for addressing said function tables and, based upon said **approval** rules, for determining the **approval** path among the system attached users; and,
- means sensitive to said **approval** path determination for monitoring the mailing and processing of said filled-in form accordingly.

2. A system according to claim 1 wherein said means for... ..and means for linking said tables together in a tree shaped arrangement.

3. A system according to claim 2 wherein said means for determining the **approval** path include:

- means sensitive to said **approval** rules for reading document data by addressing specific SQL data tables;
- means sensitive to said data read to address said function tables and fetch **approvers** references therefrom;

- means sensitive to said stored **approval** rules for listing said **approvers** references in a predefined sequential order into an **approval** list; and,
- means for storing said **approval** list into a FUTU table.

4. A system according to claim 3 further including means sensitive to read data for generating an additional list of... ..user;

- means sensitive to said stored rules for enabling said originating user to check and amend said FUTU table; and,
- means for initiating the document **approval** process upon completion of said checking.

6. A system according to claim 5 wherein said means for initiating the document **approval** process include:

- means for loading first **approver** in FUTU table into a NEXTWAIT table and loading remaining FUTU table contents into a NEXTFUTU table; and,
- means sensitive to NEXTWAIT contents to mail a predefined message to the corresponding **approver's** VM machine.

7. A system according to claim 6 including:

- means for unloading said DONE table into a NEXTDONE table within VM user's.... ..NEXTDONE user's tables into SQL tables APPFUTU, APPWAIT and APPDONE respectively.

8. A system according to claim 7 wherein said means for managing appropriate **approvals** include:

- user's terminal controllable means for requesting access to system SQL tables;
- system controllable means for accessing stored function tables, comparing terminal user's identification to **approver's** as stored into said function tables and, upon positive check, unloading contents of predefined fields of APPFUTU, APPWAIT and APPDONE SQL tables into user.... ..and DONE respectively;
- system controllable means sensitive to said user's machine tables contents for displaying preselected data from documents waiting for said user's **approval**;
- user's terminal controllable means for selecting one of said documents whereby said selected document is being displayed to said user; and
- , - terminal controllable means for said user inserting **approval** or disapproval decision (signature) into a predefined document field.

9. A system according to claim 8 wherein said means for inserting **approval/disapproval** user's decision include system controlled signature validation means and, upon correlative positive validity check, means sensitive to said validation means for writing user...

**Claims: ...B1**

1. A system for controlling the processing and routing of a user originated document requiring electronic **approval** by selected system users, in an electronic mailing system including terminals (T1, T2, T3) attached to a digital network, user's Virtual Machines (VM) including...  
...system attached users, and software controlled computer means for generating, processing and monitoring electronic documents to be mailed from any terminal to any user for **approval**, said **approval** system including means for scheduling **approval** tasks for users and means for defining a path or route specification for any document to be approved by various users, said means for scheduling **approval** and specifying a path being characterized in that it includes :

first storage means (SEALDBA Machine; Function Tables) and terminal controllable means for storing and updating... ...Structured Query Language (SQL) form, and software operated computer means for linking said tables together in a tree-shaped arrangement ;

third storage means (SEALDBA Machine **Approval** Tables) for storing predefined **approval** rules based on user's function and document forms contents ;

terminal controllable means (User's Virtual Machine) for selecting, accessing, filling-in, processing and mailing any selected form whose contents is to be subjected to **approval**;

first software controlled computer means sensitive to said mailing for addressing said function tables and, based upon said **approval** rules, for **dynamically** determining the **approval** path or route among the system attached users, said means for determining the **approval** route including :

second software controlled computer means sensitive to said **approval** rules for reading document data by addressing specific SQL data tables;

third software controlled computer means (FQMANAD1, FaMANAD2) sensitive to said read data to address said function tables and fetch **approvers'** references therefrom ;

fourth software controlled computer means sensitive to said stored **approval** rules for listing said **approvers'** references in a predefined sequential order into an **approval** list ; and,

fourth storage means for storing said **approval** list into a table (FUTU) whereby said **approval** path is determined and used for monitoring the mailing and processing of said filled-in form accordingly.

2. A system according to claim 1 further... ...said stored rules for enabling said originating user to check and amend said FUTU table ; and,

seventh software controlled computer means for initiating the document **approval** process upon completion of said checking.

4. A system according to claim 3 wherein said means for initiating the document **approval** process include :

eighth software controlled computer means for loading first **approver** in the FUTU table into a prestored NEXTWAIT table and loading remaining FUTU table contents into a prestored NEXTFUTU table ; and,

ninth software controlled computer means sensitive to NEXTWAIT table contents to mail a predefined message to the corresponding **approver's** VM-machine.

5. A system according to claim 4 including :

tenth software controlled computer means for unloading said DONE table into a prestored NEXTDONE ... ..user's tables into prestored SQL tables APPFUTU, APPWAIT and APPDONE respectively.

6. A system according to claim 5 wherein said means for managing appropriate **approvals** include :

user's terminal controllable means for requesting access to system SQL tables ;

system controllable means for accessing stored function tables, comparing terminal user's identification to **approver's** as stored into said function tables and, upon positive check, unloading contents of predefined fields of APPFUTU, APPWAIT and APPDONE SQL tables into user... ..and DONE respectively ;

system controllable means sensitive to said user's machine tables contents for displaying preselected data from documents waiting for said user's **approval** ;

user's terminal controllable means for selecting one of said documents whereby said selected document is being displayed to said user ; and

terminal controllable means for said user inserting **approval** or disapproval decision (signature) into a predefined document field.

7. A system according to claim 6 wherein said means for inserting user's decision include system controlled decision (**approval**) validation means and, upon correlative positive validity check, means sensitive to said validation means for writing user's decision into SQL table APPDONE otherwise accessible on a read-only basis.

8. A system according to claim 7 wherein said **approval** validation means include software controlled computer means for fetching a system prestored SQL command and means sensitive to said command for triggering said **approval** validity check.

9. A system according to claim 8 wherein said **approval** validation means include :

software controlled computer means for addressing the APPWAIT system table and checking presence of the considered document therein; and,

software controlled computer...

**? s s8 and ((dynamic or dynamically)(4n)(approve or approves or approved or approver or approvers or approval or approvals or approving)(4n)(list or lists or listing or listigs))**

```

      14  S8
329270  DYNAMIC
118056  DYNAMICALLY
      7019 APPROVE
      4127 APPROVES
      79727 APPROVED
      658  APPROVER
      211  APPROVERS
33891  APPROVAL
      2001 APPROVALS
      3694 APPROVING
325166  LIST
      97253 LISTS
114405  LISTING
      0    LISTIGS
      5    (DYNAMIC OR DYNAMICALLY) (4N) ((((((APPROVE OR APPROVES)
OR APPROVED) OR APPROVER) OR APPROVERS) OR APPROVAL) OR
APPROVALS) OR APPROVING) (4N) (((LIST OR LISTS) OR LISTING)
OR LISTIGS)
S9      1  S8 AND ((DYNAMIC OR DYNAMICALLY) (4N) (APPROVE OR APPROVES
OR APPROVED OR APPROVER OR APPROVERS OR APPROVAL OR
APPROVALS OR APPROVING) (4N) (LIST OR LISTS OR LISTING OR
LISTIGS))
```

**? t s9/3,k/all**

DIALOG(R)File 348: EUROPEAN PATENTS

(c) 2011 European Patent Office. All rights reserved.

9/3K/1 (Item 1 from file: 348)

00836626

### **Method and apparatus for distributing conditional work flow processes among a plurality of users**

Verfahren und Vorrichtung zum Verteilen von konditionellen Arbeitsflussprozessen zwischen mehreren Benutzern

Methode et appareil pour la distribution de processus conditionnel de flux de travail entre plusieurs utilisateurs

**Patent Assignee:**

- **Dun & Bradstreet Software Services, Inc.** (2047260)  
3445 Peachtree Street, NE; Atlanta, Georgia 30326-1276 (US)  
(applicant designated states:  
AT;BE;CH;DE;DK;ES;FI;FR;GB;GR;IE;IT;LI;LU;MC;NL;PT;SE)

**Inventor:**

- **Rossi, Charles**  
1 Indian Meadow Drive; Northborough, Massachusetts 01532; (US)
- **Vinter, Stephen**  
23 Hundred Oaks Lane; Ashland, Massachusetts 01721; (US)
- **Conte, Leonard**  
281 Green Street; Northborough, Massachusetts 01532; (US)
- **Chang, S. Jay**  
3 Duggan Road; Acton, Massachusetts 01720; (US)
- **Botzer, Robert**  
49 Delmar Avenue; Framingham, Massachusetts 01701; (US)
- **McAllister, Sandra**  
28 Lee Street; Lancaster, Massachusetts 01523; (US)
- **Dorden, Joanne**  
2 Nottingham Road; Grafton, Massachusetts 01519; (US)

**Legal Representative:**

- **Brunner, Michael John (28871)**  
GILL JENNINGS & EVERY Broadgate House 7 Eldon Street; London EC2M 7LH;  
(GB)

	Country	Number	Kind	Date	
Patent	EP	774725	A2	19970521	(Basic)
Patent	EP	774725	A3	19981028	
Application	EP	96304925		19960703	
Priorities	US	557531		19951114	

**Designated States:**

AT; BE; CH; DE; DK; ES; FI; FR; GB; GR;  
IE; IT; LI; LU; MC; NL; PT; SE

**International Patent Class (V7):** G06F-017/60; ;

**Abstract** ...used to determine what the next step in the workflow should be, to determine who the next step should be assigned to, to select which **approvers** on an **approval** list are used, etc.

Various types of conditional comparisons may be made in order to perform this functionality. Yet another feature of the present invention...

**Abstract Word Count:** 115

**Language** Publication: English

Procedural: English

Application: English

Fulltext Availability Available Text	Language	Update	Word Count
CLAIMS A	(English)	EPAB97	946
SPEC A	(English)	EPAB97	41563
Total Word Count (Document A) 42509			
Total Word Count (Document B) 0			
Total Word Count (All Documents) 42509			

**Specification:** ...work flow model.

An example of such a work flow is the routing of a document within an organization for the purpose of obtaining an **approval**. In a computer system for implementing a **purchase order** flow, a first user, such as a purchasing agent, initially creates a **purchase order** document. The creation of the **purchase order** document may include adding **purchase order** information to a computerized **purchase order** document image, drafting a **purchase order** computer document or a combination of both. After the **purchase order** document is created and at the command of the purchasing agent, the document may then be electronically routed to the second user, such as the purchasing manager, who could simply **approve/disapprove** the document or may add, delete or change information in the document prior to giving his/her **approval**. Finally, the document may be electronically routed to a third user, such as a finance or engineering manager, who may either **approve** or disapprove the **purchase order** document. The document may then be electronically routed back to the purchasing agent, who may then act upon the document accordingly.

A drawback associated with... the quantity and cost of the part. Nonetheless, both the original paper order system and its computerized implementation provide all of the information in the **purchase order** to everyone, rather than only the information relevant to each individuals specific needs. Where large documents are involved, each individual may be then forced to...used to determine what the next step in the workflow should be, to determine who the next step should be assigned to, to select which **approvers** on an **approval** list are used, etc. Various types of conditional comparisons may be made in order to perform this functionality.

Yet another feature of the present invention... activities/tasks 250 represented as messages 750 available to this user in his or her "New To Do List" To Do List window 700 are " **Approve** class registrations", "Registration confirmation," and "Select payment type for ...next activities/tasks 250 to the left of each message 750. In this example, the To Do list window 700, to the left of the "**Approve** class registrations" message 750, reveals that there are eight next

activities/tasks 250 for this category, where six are new 770 and two are done... ..undertaken in this flow process. For this example, the next steps 230 are "Review Part Planning information" to be done by the manufacturing manager and "**approve** part planning" (not shown) to be done by the quality department manager.

The "review part planning information" message 750, representative of a next activity/task...alphabetical order where no sequence specific information for an activity is indicated.

For this example, the user has selected the Class Registration, Class Payment, Registration **Approval**, and Activity activities 1760 for his or her "Sample Class Registration" list 1750. Moreover, the user has chosen to sequentially list the activities such that the Class Registration Activity has the highest SEQ(underscore)NBR, with Class Payment and Registration **Approval** having lower SEQ(underscore)NBRs and Activity having the lowest or no SEQ(underscore)NBR.

From the "Sample Class Registration" list window 1750, the user...see FIG. 12). Other next steps selected by the administrator include the "Class payment type selected" (pamsam2up1) event with the next activity/task being "Registration **Approval**" (pam0520) which is assigned to user RSD.

The administrator may also activate the archiving feature of the computer system of the present invention. This feature...230 of FIG. 2), to determine to whom the next step should be assigned (e.g., element 240 of FIG. 2), or to select which **approvers** on an **approval** list should be used. In one embodiment, the following operations may be used as conditional logic:

- \* Comparing an integer or numeric column to a constant... element 220 of FIG. 2) within conditional logic. An enhanced trigger event function may be implemented to allow applications to pass more than 16 values.

**Approval Lists.** Adding **approval** lists to the present invention adds the following functionality:

- \* Provide **approve** and reject as activity actions. This allows the **approver** to see the object that they are **approving**. Otherwise, **approvers** in a work flow use an **approval** window and have to zoom to the actual object.

- \* Provide a consistent handling of **approvals**. This ensures that all the **approval** windows have the same look and feel. It also decreases the development cost of adding **approvals** to new windows and the maintenance cost of the **approval** maintenance windows.

- \* Automatic support for new **Approval** features as they are added to the platform. These include substitutes, roles, and conditional generation.

- \* Allow users to **approve** items directly from the Task Details list (element 5601 of FIG. 56) without actually entering the application activity.



Structures Integration. The present invention may be...replace the previous workflow workbench with a new user interface that is more intuitive and which supports specification of conditional next steps, conditional assignments, and **approvals**.

Mail Integration. The present invention may be designed to support mail integration by creating an "attachment". Mail would be used as the delivery mechanism for...Data 10010, Workflow Definition 10015, and Message Queue tables 10050. The Workflow Engine 10045 may also include logic to resolve roles and substitutes, defined previously.

**Approvals** may be built into the workflow. **Approvals** build on the application specific **approval** processes that are already in place, and require the application developer to do the following:

- 1) Add an **approval** table in the application database. The key of **approval** table is the application table key, a sequence number, and an owner. The entries with the key matching the key of the application table row would be the **approval** list for that row.
- 2) The platform provides **approval** lists. When an **approval** is the next step the **approval** list to be used is passed to the **approval** window as data. Conditional logic can be supported in the determining of which **approval** list is to be used. **Dynamic** generation of the **approval** list will allow the name of the **approval** list to be used to be a field on the application window.
- 3) The platform will also provide a generalized window for tracking **approvals**. This window will be the management interface into the application provided **approval** table. The **approval** tracking window may either be an ancestor window that the applications use to create their own descendant tracking window with the proper keys or may...id or workgroup id of the substitute user or workgroup.

\* subst(underscore)owner(underscore)type - Indicates whether subst(underscore)owner is a user or workgroup. **approval(underscore)list**. The **approval** list table contains the header information for each **approval** list. It contains the following columns:

\* apprvl(underscore)list(underscore)id - The name of the **approval** list.

\* apprvl(underscore)activity(underscore)id - The activity the **approval** list is defined for.

\* notes - Notes that the user can use to describe an **approval** list. **approval(underscore)list(underscore)detail**. The **approval** list detail table contains a line item for each **approver** on the list. It contains the following columns:

\* apprvl(underscore)list(underscore)id - The name of the **approval** list.

\* apprvl(underscore)activity(underscore)id - The activity the **approval** list is defined for.

\* apprvl(underscore)level - The level for this **approver** on the list.

- \* `apprvl(underscore)id` - The user ID, workgroup or role of the **approver**.
- \* `apprvl(underscore)type(underscore)code` - The type of `apprvl(underscore)id`, either user (u), workgroup (g), role (r), or structure function (f).
- \* `structure(underscore)function...workflow engine` will have to resolve the role.

Substitute processing - If the user has a substitute defined assign change the assignee to the substitute user.

**Approvals** -- **Approvals** are described in further detail in a later section. The new algorithm for the `psp(underscore)trigger(underscore)ams(underscore)event` stored procedure is described...  
...tables and return a boolean TRUE or FALSE to indicate the value of the expression.

Conditional workflow will preferably require the following application architecture changes:

**Approve, Reject.** **Approve** and **Reject** are few actions that are only available for windows that support **approvals**. **Approve** and **Reject** will execute the logic that is currently in the **approval** windows. These actions are described in further detail in a later section.

Fix step completion bugs. The logic within basic window which controls when a... ..postfix notation before storing them in the appropriate conditional table. This work flow workbench mapping tool is defined in further detail in a later section.

**Approval Tracking.** This window may be keyed by the application key. It may display the current status of the **approval** process for the item shown. It also shows a list of the future **approvers**. **Approvals** are described in further detail in a later section.

**Approval List Definition.** This window allows the administrator to create an **approval** list. Each list is keyed by the list name and the activity class for which it is applicable. The global activity class (\*) means that the **approval** list can be used for any window in the system that supports **approvals**. **Approvals** are described in further detail in a later section.

**Role Definition.** The Role Definition window is a simple tabular window used to maintain the role...structure functions as the assign to in next step options requires that structures be enhanced to include work flow assignee fields within the point definition.

## Approvals

### Definitions

**Approval Enabled Activity** - A window that supports user configuration of **Approval** workflow by calling the new **Approval** `trig(underscore)event` function and allowing for the possibility that no **Approval** process was required. The activity is identified to the workflow mapping tool through the new `apprvl(underscore)enabled(underscore)activity` table.

**Approval Enabled Event** - An event that allows users to map a workflow using an **Approval List** for distribution of Next Step messages.

**Approval List** - A list of Users, Workgroups, Roles or structures functions which supports sequential grouping used in an **Approval** process for an **Approval Enabled Activity**.

**Approval Tracking/Status** - An application window/table that supports tracking of a single instance of the **Approval** process and access to **Approval** comments. Shows **Approval** level, **approver**, name and type, **Approval Status**.

**Substitutes** - A User or Workgroup that will temporarily receive Workflow messages intended for a specified User. This will apply to ALL Workflow not just **Approvals**.

**Workflow Mapping Tool** - A completely new graphical user interface for defining and describing workflows. It is intuitive and supports specification of conditional Next Steps, conditional assignments and assignment of **Approval Lists**. It is described in further detail below.

The main goal of the **Approvals** portion of the present invention is to provide the user 120 with a way to customize the **Approval** process with the least amount of work required by the applications. A secondary goal is to provide a model, with supporting tools, to standardize the way we do **Approvals**. This will allow automatic support of new **Approval** features as they are added to the platform. These include Substitutes, Roles, and conditional generation

Four application windows which use an **Approval** process have been used as the basis for the design. Currently, each transaction window that supports **approvals** has:

- 1) An **Approval List** table, stored procedures and a maintenance window with a "Copy From" response window.
- 2) An **Approval Substitutes** table, stored procedures and a maintenance window.
- 3) An **Approval Tracking/Status** table to monitor the **approval** process of a specific instance, a comments table, stored procedures and a maintenance window.
- 4) Some mechanism for either **approving** or rejecting and handling the updates to tracking/status tables.
- 5) Scripts to read the instance table, generate the appropriate To Do messages and to recognize when the **Approval** process is complete.

Payment Request and Journal Entry **Approvals** are very similar in that they both use the common **Approval List/Copy From** and Substitutes ancestor windows and have an **Approval Status** table with associated Comments. Purchase Requisitions use a similar format for Lists and Substitutes but have these tables in a different table family than the Status and Comments tables. They also support **Approval** at both the line and the document level. All three use the **Approval**

Tracking window as the Next Step activity in the **Approval** process, thus making the object being approved only visible at **Approval** time through Zoom. Requisitions provides an **Approval** view on the Requisition maintenance window. ECR/ECN seems the most different from the others with Comments associated with the document being approved rather than the Status table, **Approval** from the document window and different column names and datatypes for Lists and Substitutes.

To provide **Approval** Enabled activities, i.e. activities which support user modification of the **Approval** process through the use of the Workflow Mapping Tool, and to integrate new workflow features in to that process, the following tasks must be accomplished:

- 1.) Provide a mechanism to identify **Approval** Enabled activities and events and their components, e.g. the **Approving** activity. Also, indicate when a To Do is for an **Approval** activity.
- 2) Consolidate multiple application **Approval** List tables into a single platform table. Add support for structures based functions that can be used within workflow to determine workflow assignees. Structures which... ..the assignee by getting the workflow dynamic properties of the ancestor of a point at a given layer.

Determine the assignee by getting the workflow **dynamic** properties of a point.

- 3) Incorporate **Approval** List Substitutes into ...is defined, To Dos that are already in the users To Do list are not affected
- 4) Provide a model window for consistent handling of **Approval** Status/Tracking. This would make sure that all the **Approval** Tracking windows have the same look and feel. It would also decrease the development cost of adding **Approvals** to new windows. The existing application tracking windows can still be used. The GUI is generally consistent.
- 5) Provide a PowerBuilder object/ancestor to support **Approve**, Reject and Comments from within an application window which will allow a view of the object being approved. This change would allow the **approver** to see the object that they are **approving**. Currently, **approvers** use the **Approval** Tracking window and have to Zoom to the actual object. For this release, the view of the actual object will be read-only.
- 6) Allow users to **Approve** items directly from the Task Details list without actually entering the application activity.
- 7). Provide workflow functions which will consolidate code so that an **Approval** Enabled window can use two functions to do ALL of the work required including

Generating rows for the tracking/status instance table for each **approver** on the user selected **approval** list

Generating To Dos for the first level of **approvers**

Generating any additional To Dos for Next Steps attached to the **Approval** enabled event

Returning the ID and the type of the **approver** used or an indicator that no **approval** process was required

New workflow functions may be written to support **Approvals**. Both are a variation of `pam0011(underscore)trig(underscore)event()`. The first, `pam0012(underscore)trig(underscore)event()`, will return a code indicating **Approval** Complete and will require three (3) additional arguments, `tracking(underscore)SQL(underscore)string`, `approver(underscore)used` and `approver (underscore)type(underscore)used`.

The flow of the new function will be:

- \* Do the same things that the old `trig(underscore)event` function does PLUS
- \* If the **approval** flag is set in a `next(underscore)step` then evaluate the `next(underscore)step(underscore)options` conditions to get the **Approver** ID and `approver(underscore)type(underscore)code`.
- \* If no **approval** is required, i.e. no list value is present, return 1, an indicator that the **Approval** is complete.
- \* Otherwise...

Update the `approver(underscore)used` and `approver(underscore)type(underscore)code` arguments for return to the caller .

- \* Using the application window provided string containing `svr(underscore)db(underscore)owner.stored(underscore)proc` and the formatted key values, insert rows into the Tracking table. Resolve Structures functions to provide the User/Workgroup/Role ID and the **Approval** Level for each **approver**. Set the status to N.

The second, `pam0013(underscore)notify(underscore)approvers()`, will generate the next batch of **Approval** To Dos as specified by the `apprvl(underscore)level` and will require one (1) additional argument, `next(underscore)approver (underscore)SQL(underscore)string`. It can also be used in the **Approval** window.

With the prior system, when an **approval** process is associated with an activity:

- \* From `dbupdate` event, call `update(underscore)insert(underscore)tran()` to insert a row into the database for the object to be approved.
- \* error handling

call `pam0011(underscore)trig(underscore)am(underscore)event(insertevent,... )`

select the owners from the **approval** list

build a SQL(underscore)string for each **approver** for the insert stored procedure for the instance table

insert a row into the instance table for each **approver**

call pam0011(underscore)trig(underscore)am(underscore)event(sendapprmsg,.... ) for each **approver** in the first batch.

\* Using the new functions, call update(underscore)insert(underscore)tran() to insert a row into the database for the object to be approved

error handling

complete(underscore)var = pam0012(underscore)trig(underscore) **approval**(underscore)event  
(insertevent ,...,SQL(underscore)string, **approver**(underscore)used,  
**approver**(underscore)type(underscore)code) where SQL(underscore)string =  
am(underscore)server(underscore)db(underscore)owner(underscore)g +  
insert(underscore)instance(underscore)stored(underscore)procedure + &  
instance(underscore)keys (formatted for SQL)

pam0013(underscore)notify(underscore)**approvers**(sendmsgappr, .. .. ,SQL(underscore)string,  
**approval**(underscore)level) where SQL(underscore)string =  
am(underscore)server(underscore)db(underscore)owner(underscore)g +  
date(underscore)instance(underscore)stored(underscore)procedure + & instance-keys  
(formatted... ..the values for the tracking/status table are the last parameters passed in.

The description below provides additional guidance and notes in the implementation of **Approvals**.

### **Approval Table Design**

The following tables change in the present invention to support **Approvals**:

next(underscore)step

\* The apprvl(underscore)ind column indicates whether the next(underscore)step is for an **Approval**. The mapping tool and the workflow engine handle **Approvals** slightly differently from other To Dos (see below).

message(underscore)queue(underscore)1

\* The apprvl(underscore)ind column is added to indicate that the To Do is an **Approval** To Do. The Task Detail window will allow **Approval** from there if TRUE.

The following tables are added :

apprvl(underscore)enabled(underscore)activity

The apprvl(underscore)enabled(underscore)activity table contains the IDs of Activities that are **Approval Enabled**. It contains the following columns:

- \* activity(underscore)id (primary key) - The ID of an activity that has been coded to support customized **Approvals**.
- \* apprvl(underscore)event(underscore)id (primary key) - The ID of an event which can generate an **Approval To Do**.
- \* reapprvl(underscore)event(underscore)id - The ID of an event which will require reapproval because of changes to the object being approved.
- \* apprvl(underscore)activity(underscore)id - The ID of the activity where the **Approval** should be performed, i.e. the Next Step activity.
- \* task(underscore)detail(underscore)apprvl(underscore)id - The name of the PowerBuilder object that contains the application code to be used by Task Detail to support Approvals for this activity.

apprvl(underscore)list

The apprvl(underscore)list table contains **Approval** List IDs associated with the activities that use them. It contains the following columns:

- \* apprvl(underscore)list(underscore)id (primary key) - The ID of an **Approval** List.
- \* apprvl(underscore)activity(underscore)id (primary key) - The ID of an activity that has access to this list for **Approvals**.
- \* notes - Text used as needed.

apprvl(underscore)list(underscore)detail

The apprvl-list(underscore)detail table contains a list of users, groups, roles or structures functions which can be associated with an **Approval** process. It contains the following columns:

- \* apprvl(underscore)list(underscore)id (primary(underscore)key) - The ID of the **Approval** List whose members are defined here.
- \* apprvl(underscore)activity(underscore)id (primary key) The activity that has access to this list.
- \* apprvl(underscore)level (primary key) - Indicates the order in which **Approvals** are performed.

- \* apprvr(underscore)id (primary key) - The user, workgroup or role ID of this **approver** or the default **approver** should a structures function fail to be resolved.
- \* apprvr(underscore)type(underscore)code (primary key) - Indicates whether **approver(underscore)id** is a user, workgroup, role or structure function.
- \* structure(underscore)function(underscore)type - Indicates the type of structure being used.
- \* structure(underscore)group... ..column ID of the data being used to resolve this structure function.
- \* layer(underscore)name - The name of the Layer.
- \* default(underscore)type - Indicates whether **approver(underscore)id** is a user, workgroup, role. Relevant when the structure function cannot be resolved and the **approver(underscore)id** is used as a default.

#### substitute

The Substitute tables contains a list of users who have substitutes defined for them. It contains the... ..number and datatype of the keys will vary from application to application.

- \* instance key(s) (primary key)
- \* apprvl(underscore)level (primary key)- Order in which **approvals** are performed.
- \* apprvr(underscore)id (primary key)- The user, workgroup or role ID of this **approver**.
- \* apprvr(underscore)type(underscore)code (primary key) - User, Workgroup or Role
- \* apprvl(underscore)status(underscore)code - N = None (hasnt been sent a message to review... ..Date of the last status change
- \* actual(underscore)apprvr(underscore)id - ID of the user who performed the status update.

#### End User Windows

##### Activity Window **Approve/Reject** Standard

The new activity will inherit from the original activity to provide a complete, read-only view of whatever is being Approved. Functionality includes **Approve**, Reject, Comments, **Approval** Complete, generation of next level To Do messages and updates to the instance tracking table. The Yellow Sticky object is moveable so that all portions... ..seen.

#### Task Detail



plt0040(underscore)todo needs to be modified so that when the message queue row indicates that the Next Step activity is an **Approval**, an additional button will be displayed. When rows from the detail list are selected, enable the button. If **Approval** button clicked, execute the application provided code to update the **Approval** Tracking, generate trig(underscore)events and complete the **approval** process as needed. A reusable PowerBuilder object will contain the necessary application code.

#### User Substitute

The Substitute window allows a user to define a substitute... ..with the system so that an informational To Do is generated when a user is made a substitute for another user.

#### Administration and Tracking Windows

##### **Approval** List Definition

This window allows the administrator to create an **Approval** List. Each list is keyed by the list name and the activity ID for which it is applicable. A Popmenu and response window support Copy from one List to another.

##### Tracking/Status Ancestor

This window is keyed by the application key(s). It shows the current status of the **Approval** process for the item shown. It also shows a list of the future **approvers** and allows the current user to **Approve** or **Reject** if appropriate. PopMenu and Tool Bar provide Accept, Reject and Zoom to Document actions.

##### Administrator Substitute

The Administrator Substitute window is a simple... ..are assign(underscore)to type and user or workgroup.

Rule: Add, change, delete of Substitutes does NOT impact existing workflow.

##### Workflow Mapping Tool

When an **Approval** Enabled activity is mapped, the map knows which events are **Approval** Enabled. When the Insert event is chosen from the windows list of available events, the **Approval** Activity is presented as a possible next step. When chosen the windows presentation is slightly different in that **Approval** Lists are presented for the assignment of To Do messages.

##### Application Impact Issues

**Approval** Enabled activities must:

1) Have moved **Approval** Lists and Substitutes to the common Platform tables. Note: Substitutes will now apply to all workflow NOT just **Approvals**. These tables and their respective maintenance windows will replace the applications' individual implementations. This means that two windows can be dropped for each **Approval** Enabled activity.

2) Use the new TrigEvent function to manage **approval** processing and to process "**Approval Complete**" code, if the user has chosen to NOT **approve** under certain conditions. This function will return an indicator of **Approval** Complete and the ID of the **Approver** and its type code which was used to generate the Tracking Table rows. The application window will provide a string containing svr(underscore)db(underscore)... ..proc and the formatted key values to Insert rows into the Tracking table. The workflow engine will provide the User/Workgroup/Role ID and the **Approval** Level for each **approver**. Structures functions will be resolved at this time. This means that some code can be removed from an activity window. Each application designer should consider whether it would be useful to store the name of the **Approver** used.

3) Have written the PowerBuilder function which will process **Approvals** from the Task Detail window. A model will be provided and an attempt will be made to require the bare minimum of code from the... ..the-blanks with code that currently exists in other places.

Other steps which could be taken to move in the direction of an internally consistent **Approval** process are:

1) Rewrite Tracking/Status windows to use new standard.

2) Use the model for **Approve** /Reject from the document in addition to the **Approve** from the Tracking window.

StreamBuilder Impact Issues

**Approval** Activity window

Users who build activities which they wish to **Approval** Enable will need a GUI to enter and maintain the apprvl(underscore)enabled(underscore)activity table. **Approval** Enable one of the existing Sample Application windows. (psa0800(underscore)invoice). Use the new Yellow Sticky to do the **Approval** on a read-only view of the original activity using the new model. (psa0810(underscore)invoice(underscore)**approval**).

**Approval** Tracking Window

Provide a sample of the **Approval** Tracking/Status implementation using the new Ancestor. (psa0850(underscore)invoice(underscore)status).

Task Detail **Approval** object

Provide a sample of the PowerBuilder function used to **Approve** from the Task Detail window. (psa0800(underscore)invoice(underscore)**approve**).

## Dependencies

1) Structures integration requires that all structures that are used within workflows be replicated to all work flow servers 110 in the system.

2...

? ds

Set	Items	Description
S1	9917694	PD<20021017
S2	0	AU=(CIRULLI, S OR CIRULLI S? OR (SUSAN(2N)CIRULLI)) OR BY=(SUSAN(2N)CIRULLI)
S3	28	AU=(HAGER, D OR HAGER D? OR ((DAN OR DANNY) (2N)HAGER)) OR - BY=((DAN OR DANNY) (2N)HAGER)
S4	0	S1 AND (S2 OR S3) AND (APPROVE OR APPROVER OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING OR SIGNOFF) AND (PURCHASE(-W) (FORM OR FORMS OR ORDER OR ORDERS OR REQUISTION OR REQUISTIONS))
S5	0	S1 AND ((DAN OR DANNY) (2N)HAGER) OR (SUSAN(2N)CIRULLI)
S6	10391056	PD<20031017
S7	517	S1 AND (APPROVE OR APPROVER OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING OR SIGNOFF) AND (PURCHASE(W) (FORM OR FORMS OR ORDER OR ORDERS OR REQUISTION OR REQUISTIONS))
S8	14	S7 AND ((DYNAMIC OR DYNAMICALLY OR RECALCULATE OR RECALCULATES OR RECALCULATED OR RECALCULATING OR RECALCULATION OR RECALCULATIONS) (4N) (APPROVE OR APPROVES OR APPROVED OR APPROVER - OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING))
S9	1	S8 AND ((DYNAMIC OR DYNAMICALLY) (4N) (APPROVE OR APPROVES OR APPROVED OR APPROVER OR APPROVERS OR APPROVAL OR APPROVALS OR APPROVING) (4N) (LIST OR LISTS OR LISTING OR LISTIGS))

?

## ? logon

\*\*\* It is now 2011/11/21 06:45:35 \*\*\*

(Dialog time 2011/11/21 06:45:35)

705TEXT1 is set ON as an alias for 15, 16, 160, 148, 621, 275, 634, 47

705TEXT2 is set ON as an alias for 9, 623, 810, 624, 813, 20, 636

705BIBLIT is set ON as an alias for 77, 35, 583, 2, 65, 233, 99

705NEWSBIB is set ON as an alias for 473, 474, 475

SOFTLIT is set ON as an alias for 256, 278

705ADLIT is set ON as an alias for 635, 570, PAPERSMJ, PAPERSEU

HIGHLIGHT set on as ' ' ' '

DETAIL set off

KWIC is set to 50.

FTEXT1 is set ON as an alias for 15,9,610,810,275,634,471

FTEXT2 is set ON as an alias for 613,813,636,16,160,621,148,20,624

FTPAT is set ON as an alias for 348,65

NFTEXT is set ON as an alias for 35,65,99,2,583,474,475,347,256

## ? b 347,348,349

21nov11 06:48:22 User264751 Session D860.1

\$0.00 0.249 DialUnits File415

\$0.00 Estimated cost File415

\$0.82 INTERNET

\$0.82 Estimated cost this search

\$0.82 Estimated total session cost 0.249 DialUnits

SYSTEM:OS - DIALOG OneSearch

File 347:JAPIO Dec 1976-2011/JULY(Updated 111028)

(c) 2011 JPO & JAPIO

File 348:EUROPEAN PATENTS 1978-201146

(c) 2011 European Patent Office

File 349:PCT FULLTEXT 1979-2011/UB=20111110|UT=20111103

(c) 2011 WIPO/Thomson

Set	Items	Description
-----	-------	-------------

---	-----	-----
-----	-------	-------

## ? t 00836626

>>> Direct TYPE is available "FROM" a single file only

## ? t 00836626/9

>>> Direct TYPE is available "FROM" a single file only

## ? b 348

21nov11 06:49:23 User264751 Session D860.2

\$1.47 0.121 DialUnits File347

\$1.47 Estimated cost File347

\$0.73 0.121 DialUnits File348

\$0.73 Estimated cost File348

\$0.62 0.121 DialUnits File349

\$0.62 Estimated cost File349

OneSearch, 3 files, 0.364 DialUnits FileOS

\$0.54 INTERNET  
 \$3.36 Estimated cost this search  
 \$4.18 Estimated total session cost 0.612 DialUnits

File 348:EUROPEAN PATENTS 1978-201146  
 (c) 2011 European Patent Office

Set	Items	Description
---	-----	-----

? t 00836626/9

DIALOG(R)File 348: EUROPEAN PATENTS  
 (c) 2011 European Patent Office. All rights reserved.  
 836626/  
 00836626

# **Method and apparatus for distributing conditional work flow processes among a plurality of users**

Verfahren und Vorrichtung zum Verteilen von konditionellen Arbeitsflussprozessen zwischen mehreren Benutzern

Methode et appareil pour la distribution de processus conditionnel de flux de travail entre plusieurs utilisateurs

## **Patent Assignee:**

- **Dun & Bradstreet Software Services, Inc.** (2047260)  
 3445 Peachtree Street, NE; Atlanta, Georgia 30326-1276 (US)  
 (applicant designated states:  
 AT;BE;CH;DE;DK;ES;FI;FR;GB;GR;IE;IT;LI;LU;MC;NL;PT;SE)

## **Inventor:**

- **Rossi, Charles**  
 1 Indian Meadow Drive; Northborough, Massachusetts 01532; (US)
- **Vinter, Stephen**  
 23 Hundred Oaks Lane; Ashland, Massachusetts 01721; (US)
- **Conte, Leonard**  
 281 Green Street; Northborough, Massachusetts 01532; (US)
- **Chang, S. Jay**  
 3 Duggan Road; Acton, Massachusetts 01720; (US)
- **Botzer, Robert**  
 49 Delmar Avenue; Framingham, Massachusetts 01701; (US)
- **McAllister, Sandra**  
 28 Lee Street; Lancaster, Massachusetts 01523; (US)

- **Dorden, Joanne**  
2 Nottingham Road; Grafton, Massachusetts 01519; (US)

**Legal Representative:**

- **Brunner, Michael John (28871)**  
GILL JENNINGS & EVERY Broadgate House 7 Eldon Street; London EC2M 7LH; (GB)

	Country	Number	Kind	Date	
Patent	EP	774725	A2	19970521	(Basic)
Patent	EP	774725	A3	19981028	
Application	EP	96304925		19960703	
Priorities	US	557531		19951114	

**Designated States:**

AT; BE; CH; DE; DK; ES; FI; FR; GB; GR;  
IE; IT; LI; LU; MC; NL; PT; SE

**International Patent Class (V7):** G06F-017/60; ;

**Abstract** EP 774725 A2

In accordance with the teachings of the present invention, a new computerized information flow distribution technology (work flow) is provided. One feature of the present invention allows the use of conditional logic in determining how information is routed among users. Specifically, conditional logic may be used to determine what the next step in the workflow should be, to determine who the next step should be assigned to, to select which approvers on an approval list are used, etc. Various types of conditional comparisons may be made in order to perform this functionality. Yet another feature of the present invention allows the use of a graphical tool for creating and mapping the work flow processes.

**Abstract Word Count:** 115

Legal Status Type	Pub. Date	Kind	Text
Withdrawal:	20000209	A2	Date application deemed withdrawn: 19990428
Application:	19970521	A2	Published application (A1with;A2without)
Search Report:	19981028	A3	Separate publication of the European or International search report

**Language** Publication: English

Procedural: English

Application: English

Fulltext Availability Available Text	Language	Update	Word Count
CLAIMS A	(English)	EPAB97	946
SPEC A	(English)	EPAB97	41563
Total Word Count (Document A) 42509			
Total Word Count (Document B) 0			
Total Word Count (All Documents) 42509			

### Specification:

This invention relates generally to automated information flow technology and, in particular, to a system for maintaining and conditionally distributing information between a plurality of users in a transactional based information flow environment.

The use of computer systems as a means of gathering and distributing information has become commonplace in modern organizations. Prior art computer systems typically use "document based" technology to gather and distribute the organization's information in the form of computer documents stored as files.

In recent years, advancements in "document based" computer system technology have been made which facilitate the flow of information between individuals within the organization. One common procedure, known as "work flow" allows work-related information to flow among individuals within the organization. The information flows in a "document-based" form that typically follows a model of a manual work flow process, where documents are physically routed from one individual to another. Further, the "document based" computing procedure provides a means for the individuals (users of the computer system) to add, delete or change information within the computer document. The revised document is then disseminated or flowed to other users within the organization following the manual work flow model.

An example of such a work flow is the routing of a document within an organization for the purpose of obtaining an approval. In a computer system for implementing a purchase order flow, a first user, such as a purchasing agent, initially creates a purchase order document. The creation of the purchase order document may include adding purchase order information to a computerized purchase order document image, drafting a purchase order computer document or a combination of both. After the purchase order document is created and at the command of the purchasing agent, the document may then be electronically routed to the second user, such as the purchasing manager, who could simply approve/disapprove the document or may add, delete or change information in the document prior to giving his/her approval. Finally, the document may be electronically routed to a third user, such as a finance or engineering manager, who may either approve or disapprove the purchase order document. The document may then be electronically routed back to the purchasing agent, who may then act upon the document accordingly.

A drawback associated with the above prior art technology is that an entire document image, or at least all the information contained within the document, must be accessed by each individual in the information flow. This will occur even if any one of the individuals is only interested in

some, but not all, of the information in the document. For example, the purchasing manager might only be interested in information related to the quantity and vendor of a requested part, while the finance manager might only be interested in the quantity and cost of the part. Nonetheless, both the original paper order system and its computerized implementation provide all of the information in the purchase order to everyone, rather than only the information relevant to each individual's specific needs. Where large documents are involved, each individual may be then forced to sift through much information irrelevant to his or her job function.

Moreover, where documents contain sensitive information, which typically is reserved for only for specific individuals, prior art information flow systems require the creation of multiple documents with varying security authorizations and particular routing schemes for each of these documents. Thus, the prior art computerized information flow technologies are riddled with inefficiencies.

Another shortcoming associated with the above prior art technology is that the "document based" information is routed on an "ad hoc" basis. In other words, the routing of the "document based" information is controlled exclusively by the user who last added, deleted or changed information in the document. Therefore, the prior art systems rely on the users using the system to promptly and correctly route the documents to other individuals within the organization.

Limitations related to this prior art "ad hoc" technology include the possibility of users losing, misplacing, or misdirecting documents. Moreover, this prior art technology may result in time lags, procedural steps being missed, and people being left out of the process. Accountability is also a problem since many prior art systems provide no effective way to determine who is responsible for the problems caused with improper routing of a document.

Certain of the shortcomings of prior art systems have been improved upon by recent developments in work flow technology, including the inventions disclosed and claimed in co-pending U.S. Patent Application Serial No. 08/213,022, filed March 14, 1994, and U.S. Patent Application Serial No. 08/475,575, filed June 7, 1995, both commonly assigned to the assignee of the present patent application. U.S. Patent Application Serial No. 08/213,022 and U.S. Patent Application Serial No. 08/475,575 are incorporated herein by reference thereto.

U.S. Patent Application Serial No. 08/213,022 is directed to a new work flow technology which introduces a process and apparatus for facilitating the flow of information between various computer users of networked computers to complete predefined procedures within an organization. The invention of this prior co-pending patent application facilitates the flow of information by providing a unique method for logically and automatically routing information through a predefined sequence of activities to users who need the information and can act on the information. While the invention of Application Serial No. 08/213,022 represents a significant improvement over prior art work flow techniques, it is still limited in certain respects. For example, although the work flow processes using this prior invention allow users on various interconnected computers to all participate in the defined work flow, the actual work flow definitions must generally be created on a central computer system (server), resulting in the possibility of bottlenecks. Moreover, if this central computer system becomes inoperational for



some reason, the entire work flow processes risk coming to a halt, even though other computers scattered throughout the network remain operational.

U.S. Patent Application Serial No. 08/475,575 is directed to a new workflow technology that overcomes some of the limitations described above. For example, this application describes a process and apparatus for distributing work flow over a plurality of computer systems, thereby minimizing the potential for bottlenecks and allowing work flow to continue even in the presence of a failure of one or more of the servers.

Nonetheless, the previously described systems still are limited in certain respects. For example, work flow models may be created utilizing certain conditional logic, but this conditional logic is limited in its breadth. Further, the creation of the work flow model in previous systems is a somewhat cumbersome process, requiring the system administrator or developer to manually create the various dependencies within the work flow scheme.

The above illustrates just some of the problems associated with the prior art technology. These drawbacks and other shortcomings are effectively overcome by the present invention, as described in further detail below.

In accordance with the teachings of the present invention, a new computerized information flow distribution technology is provided. The present invention improves on a new method and apparatus for distributing information between a plurality of individuals with an organization, which was disclosed in co-pending U.S. Patent Applications Serial No. 08/213,022 and Serial No. 08/475,575, and which are incorporated herein by reference. Like these previous patent applications, the present invention continues to utilize "transactional based automated information flow" technology.

The technology is "transactional based" because it is based on relational database transaction technology, providing fine-grained serializability correctness for concurrent and atomic access to widely dynamic and diverse data elements stored in distributed repositories. This technology improves on the prior art "document based" technology which provides access to document-based data only. . Therefore, the "transactional based" technology improves on the "document based" technology because it provides diverse types of interrelated data to be concurrently accessible using a high-capacity, highly performant implementation.

The technology is "automated" because the information entered by one user is logically and automatically distributed through a predefined sequence to users who need the information. This technology improves on the "ad hoc" technology which only distributes information to other users designated by the user who last added, deleted or changed the information and only upon specific instructions from the user. Therefore, the "automated" technology improves on the "ad hoc" information flow technology because it allows information within an organization to be promptly, accurately, and sequentially routed to users through an information flow process predefined by the organization.

The "transactional based automated information flow" computer system of the present invention may be used in almost any organization, regardless of the location of the users within the organization. For example, users may be located across the country or even around the world.

In a preferred embodiment, the present invention may include one or more storage facilities (e.g., servers) located at each location within the organization. Further, in order to optimize the distribution of the "transactional based automated information flow" across the various storage facilities, the present invention uses replication techniques. These techniques are used to ensure that the storage facilities within the organization are periodically updated with information needed by the users of the computer system. In a preferred embodiment, the computer system includes a central storage facility, zero or more remote storage facilities, and a means for replicating data periodically. The central storage facility may be connected to the remote storage facilities through a hard-wired connection, a telecommunications link, or the like.

In one embodiment, the computer system provide Platform data which is used to provide a common set of services to applications, including workflow and security. Platform data consists of two types of data elements: replicated data, for which identical copies reside at each storage facility in the computer system, and distributed data, which is partitioned and dispersed among each storage facility.

The central storage facility is used to store original copies of replicated platform data which is generally information that all or most of the users need to access in a primarily read-only mode. Specifically, the platform data often contains administration data used by the entire organization. An organization may locate the central storage facility anywhere within the organization but typically will choose a location in close proximity to the administrator of the computer system.

The remote storage facilities are used to store read-only copies of the replicated platform data, as well as original copies of the application data. The application data is typically application programs and data that only certain users or groups of users need to access. Since the remote storage facilities store platform and application data, an organization will typically locate a remote storage facility in close proximity to the certain users who utilize the application data.

Distributed platform data is the data that is associated with each user, such as their desktop. Initially, all users and their corresponding platform data are located on the central storage facility. Tools enable administrators to move users to remote storage facilities, thereby moving their corresponding platform data to these locations. Only one copy of a user's platform data exists at a time.

The means for replicating is preferably a computer program which periodically distributes updated platform data from the central storage facility to the remote storage facilities. The preferred replicating means is the proprietary replication service provided by the database vendor. In the case of Sybase, this is the Sybase Replication Server. Additionally, an Asynchronous Remote Procedure Call (ARPC) mechanism is used to ensure reliable updates involving data in a remote or multiple storage facilities.. Whenever the platform data is scheduled for replication, the platform data from the original copy stored at the central storage facility is copied to the to each remote storage facility.

Accordingly, the present invention eliminates problems associated with many prior art computer systems in which all users within an organization access the same copy of platform data from the same central storage facility. Specifically, the "single point of failure" drawback experienced with these prior art systems is practically eliminated in that a failure at the central storage facility will have little effect on a user accessing platform data from a particular remote storage facility providing no application data from the central storage facility is needed. Further, the bottleneck problems experienced when many users of the prior art computer systems attempt to access platform data at the central storage facility at the same time is substantially diminished. This results because smaller groups of users are now able to access the platform data from particular remote storage facilities assigned to the group. Moreover, the computer time for accessing the platform data from a central storage facility at a geographically remote location in the prior art computer systems is also diminished as platform data may now be stored on remote storage facilities located in close proximity to particular groups of users.

Another feature of the present invention allows the use of conditional logic in determining how information is routed among users. Specifically, conditional logic may be used to determine what the next step in the workflow should be, to determine who the next step should be assigned to, to select which approvers on an approval list are used, etc. Various types of conditional comparisons may be made in order to perform this functionality.

Yet another feature of the present invention allows the use of a graphical tool for creating and mapping the work flow processes.

The aforementioned and other aspects of the present invention are described in the detailed description and attached illustrations which follow.

FIGS. 1A, 1E, 1F and 1H depict various configurations of a client/server network, on which the present invention may be implemented.

FIG. 2 depicts a flow diagram of the basic information flow process according to the present invention.

FIG. 3 depicts a computer screen, according to a preferred embodiment of the present invention, displaying a user's drawer, folders and lists.

FIG. 4 depicts a computer screen, according to a preferred embodiment of the present invention, displaying an activity window.

FIG. 5 depicts an example of an activity and associated events.

FIG. 6 depicts the primary information contained in a next step according to a preferred embodiment of the present invention.

FIG. 7 depicts a computer screen, according to a preferred embodiment of the present invention, displaying a user's To Do Lists in the user's To Do List folder and an exemplary user's personalized To Do list containing next activities.

FIG. 8 depicts a computer screen, according to a preferred embodiment of the present invention, displaying a user's To Do Lists in the user's To Do list folder and an exemplary work group To Do list containing next activities.

FIG. 9 depicts an illustrative example of the basic information flow process depicted in FIG. 2.

FIG. 10 depicts the structure of a table according to a preferred embodiment of the present invention.

FIGS. 11-15 depict illustrative examples of tables used in a preferred embodiment of the present invention, including the tables relationships.

FIG. 16A depicts a computer screen, according to a preferred embodiment of the present invention, displaying the File mode where the user may select the New command to create an activity list.

FIG. 16B depicts a computer screen, according to a preferred embodiment of the present invention, displaying the New Browser Objects window, where the user may name an activity list and select the location for the activity list.

FIG. 16C depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Browser mode which reveals all folders and lists for a user's drawer, where a user may select to add lists to folders.

FIG. 16D depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Browser list window, where the user may select the Customize Activity List command to add an activity to an activity list.

FIG. 16E depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Customize Activity List window, where the window reveals a list of activities the user may access.

FIG. 16F depicts a computer screen, according to a preferred embodiment of the present invention, displaying the New Browser Objects window where the user may name a To Do List and Select the location for the To Do list.

FIG. 16G depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Browser mode which reveals all folders and lists for a user's drawer, where the user may select a To Do List in which to add a next activity category.

FIG. 16H depicts a computer screen, according to a preferred embodiment of the present invention, displaying a To Do List containing a plurality of next activity categories.

FIG. 16I depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Browser list window, where the user may select the Move command to move a next activity category from the current To Do List to another To Do List.

FIG. 16J depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Move mode list window which reveals a list of possible To Do Lists for the user to choose to move a next activity category.

FIG. 16K depicts a computer screen, according to a preferred embodiment of the present invention, displaying a To Do List containing a next activity which has been moved by the user to this To Do List.

FIG. 16L depicts a computer screen, according to a preferred embodiment of the present invention, displaying a To Do List containing individual next activities, where the user may select to move an individual next activity to another To Do List.

FIG. 16M depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Browser list window, where the user may select the Move command to move an individual next activity from the current To Do List to another To Do List.

FIG. 16N depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Move mode list window which reveals a list of possible To Do Lists for the user to choose to move an individual next activity.

FIG. 16O depicts a computer screen, according to a preferred embodiment of the present invention, displaying a To Do List after an individual next activity has been moved to another To Do List.

FIG. 16P depicts a computer screen, according to a preferred embodiment of the present invention, displaying a To Do List containing an individual next activity which has been moved to the To Do List from another To Do List.

FIG. 17A depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Browser mode which reveals all folders and lists for a user's drawer, where a user may select to access a list, where the Sample Class Registration list has been selected and revealed, and where the Class Registration activity has been selected.

FIG. 17B depicts a computer screen, according to a preferred embodiment of the present invention, displaying an illustrative example of a Class Registration activity window.

FIG. 17C depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Set Task Priority option in the Options mode, where the user creating a task may set the priority of the task as low, medium or high.

FIG. 17D depicts a flow diagram of a preferred embodiment for saving information in connection with an activity, determining the event associated with the activity, and accessing the Trigger Event function stored procedure.

FIG. 18 depicts a flow diagram of a preferred embodiment of the Trigger Event function stored procedure which determines the next steps for the flow of information process, in particular next activity(s) and user(s) responsible for performing the next activity(s).

FIG. 19 depicts a computer screen, according to a preferred embodiment of the present invention, displaying an illustrative example of a second Class Registration activity window.

FIG. 20 depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Browser mode list window where the user may access a To Do List, where the "New To Do List" To Do List has been accessed, and where a list of next activity categories for the To Do List is revealed in the Summary To Do Category window.

FIG. 21 depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Browser mode list window where the user may access a To Do List, where the "New To Do List" To Do List has been accessed, and where a list of next activities for the To Do List is revealed in the Detailed To Do Category window.

FIG. 22A depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Summary To Do Category window where the "Select Payment Type for Class" next activity has been selected.

FIG. 22B depicts a computer screen, according to a preferred embodiment of the present invention, displaying an illustrative example of a Class Payment next activity window.

FIG. 22C depicts a computer screen, according to a preferred embodiment of the present invention, displaying an Options mode list accessed from the Class Registration activity window, where the user may select the Next Step command to access the next activity that occurs sequentially after the Class Registration activity and which the user is responsible for performing.

FIG. 23 depicts a flow diagram of a preferred embodiment of the Next Step procedure which determines the next activity that occurs sequentially after the just completed activity or next activity and which the user is responsible for performing.

FIG. 24 depicts a computer screen, according to a preferred embodiment of the present invention, displaying an illustrative example of a Class Payment next activity window which is the sequentially subsequent next activity after the just completed Class Registration activity and which the user is also responsible for performing.

FIG. 25 depicts a computer screen, according to a preferred embodiment of the present invention, displaying an Options mode list accessed from the Class Registration activity window, where the user may select the Next Task command to access a next activity to the computer screen based on priority settings.

FIG. 26 depicts a flow diagram of a preferred embodiment of the Next Task procedure which determines a next activity to display on the computer screen based on priority settings.

FIG. 27 depicts a computer screen, according to a preferred embodiment of the present invention, displaying an illustrative example of a Class Payment next activity which was selected by the Next Task procedure as a next activity that the user is responsible for performing.

FIG. 28 depicts a computer screen, according to a preferred embodiment of the present invention, displaying an illustrative example of a "New To Do List" To Do List containing two class payment activities, represented by the "Select payment type for Class" Category, which have been completed (done).

FIG. 29 depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Workflow Workbench mode accessed by an administrator of the computer system of the present invention which displays an activity or corresponding event so that the administrator may define information flow procedures.

FIG. 30 depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Zoom options list window, where the administrator may select an option to set up next steps in defining an information flow procedure.

FIG. 31 depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Step Assignments window which was selected from the Zoom options list window, where next steps may be assigned by the administrator.

FIG. 32 depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Step Assignments window of FIG. 31, where the administrator has partially selected next steps for the given activity.

FIG. 33 depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Options mode list, where the Zoom option has been selected so that a next activity may be selected.

FIG. 34 depicts a computer screen, according to a preferred embodiment of the present invention, displaying a To Do Category window accessed from the Zoom option, where a next activity is selected by the administrator.

FIG. 35 depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Step Assignments window of FIG. 31, where the users responsible for the next activity have been selected by the administrator.

FIG. 36 depicts a computer screen, according to a preferred embodiment of the present invention, displaying an illustrative example of a Workflow Workbench window where next steps have been defined to complete an information flow procedure.

FIG. 37 depicts a computer screen, according to a preferred embodiment of the present invention, displaying the Preferences window which allows the user to select specific features relevant to his or her To Do List, including refresh task counts at certain time intervals and notify the user when a new next activity has been added to his or her To Do List.

FIG. 38A depicts a computer screen, according to a preferred embodiment of the present invention, displaying a Summary To Do Category window, where completed next activities may be deleted manually.

FIG. 38B depicts a computer screen, according to a preferred embodiment of the present invention, displaying a Detailed To Do Category window, where completed next activities may be deleted manually.

FIG. 39 depicts a computer screen, according to a preferred embodiment of the present invention, displaying a Summary To Do Category window, where the user may obtain detailed information on a particular next activity.

FIG. 40 depicts a computer screen, according to a preferred embodiment of the present invention, displaying a To Do Informational window which reveals detailed information on a particular next activity.

FIGS. 41 through 52 depict various diagrams illustrating how the present invention may be implemented in a distributed manner.

FIGS. 54 through 74 depict various diagrams illustrating how the present invention may be implemented with conditional logic and with a graphical work flow mapping tool.

#### DETAILED DESCRIPTION OF THE INVENTION:

##### Distributed Work Flow

The computer system 100 of the present invention is preferably implemented on a client/server network 100 as shown in FIG. 1A. The client/server network 100 includes a server 110, such as an HP/UX, Data General DG-UX, Microsoft NT, IBM RS/6000, or an OS/2 server, connected to a plurality of clients 120, also known as end user workstations. Each end user workstation preferably includes a monitor 126, a screen 122, a keyboard 124, a mouse 128, and a memory device. The end user workstations 120 may be an IBM compatible PC running MS-DOS and Microsoft Windows or their equivalent. The preferred client/server network of the present invention is a Novell Netware or a PC LAN. Though these are the preferred clients, servers, and client/server networks, as may be appreciated by one of ordinary skill in the art, suitable equivalents may be used.

Referring to FIG. 2, a flow chart is shown which depicts the basic flow process for the present invention. This flow process assures that information is routed through the organization's predefined information flow path to the users who need it.

In a preferred embodiment, a user may access the computer system 100 of the present invention from an end user workstation 120 (see FIG. 1A), utilizing the particular user interface, such as the Windows Graphical User Interface (GUI), provided by the computer workstation's 120 operating system and environment. As shown in FIG. 3, the computer system of the present invention searches the user's drawer 305 to provide an Activity Lists folder 320 and a To Do List



folder 330 on the user's computer terminal screen 122. These are preferably displayed in a Browser mode window 310.

The Activity Lists folder 320 may contain one or more lists of activities that the user may select and act upon. In this example, the Activity Lists folder 320 contains a "Your Activities" To Do List 325. The To Do Lists folder 330 may contain one or more lists of tasks to be completed by the user or the user's work group. In this example, the To Do List folder 330 contains a "New To Do List" To Do List 335 and a "Finance Workgroup" To Do List 336. The Activity Lists folder 320, To Do Lists folder 330 and the contents of both folders may be displayed using a commercially available graphic user interface such as Microsoft Windows.

In this example, the user selects a list from either the Activity Lists folder 320 or To Do Lists folder 330. If the user selects the "Your Activities" list 325, the system provides a list of available activities 300 for the "Your Activities" list 325. This selection may be made, for example, by clicking over the chosen activity 210 with the mouse 128 or by cursoring over the chosen activity 210 with the tab key and hitting the return key on the keyboard 124.

In response to the user's selection of a list, the system displays a list 300 of available activities or tasks. For this example, the user's "Your Activities" list 325 contains such activities as Activity Security, Database Administration and Class Registration (English). The user may then select an activity 210 from the list 300. The system responds to the selection by displaying a screen relating to the activity 210 to be acted upon.

FIG. 4 illustrates a screen which the system displays in response to the user choosing the Class Registration (English) activity 210 from his or her "Your Activities" list 325 (see FIG. 3). The Class Registration activity is displayed in an activity window 400, which preferably reveals information in the form of headings 420 and values 430.

To illustrate, the activity window 400 for a user attempting to register for a class may reveal the headings 420 as class, student, class description and credit status. Examples of values 430 associated with the credit status heading 420 are undergraduate, graduate, and audit. The activity window 400 also typically includes prompts, also referred to as blank fields, 450. Examples of prompts 450 in the activity window 400 are shown to the left of the undergraduate, graduate, and audit values 430 under the credit status heading 420. These prompts 450 may be filled in to represent information input by the user. In this example, the student/user has filled in the prompt 450 to the left of the undergraduate value 430. In a preferred embodiment of the present invention, other information may be entered by the student/user by simply clicking over the chosen value 430 with the mouse 128 (see FIG. 1A) or cursoring over the chosen value 430 with the tab key on the keyboard 124 (see FIG. 1A).

Referring back to FIG. 2, after all of the information or data required for activity 210 has been entered by the user, the user acts to indicate to the system that the chosen activity 210 has been completed and the data supplied by the user during performance of activity 210 should be saved. In a preferred embodiment, the user may make their indication using the mouse 128 to "click over" a "save file" icon on the computer screen 122 which, in a preferred embodiment, resembles a floppy disk (not shown). Alternatively, the user may depress the keyboard "Control" and "S"

keys simultaneously (see FIG. 1A). After the data relating to activity 210 has been saved and the user indicates that the activity 210 has been completed, the system triggers one of the events 220 associated with the activity 210.

An event 220 is a representation of a set of conditions stored in the computer system in accordance with the present invention. Whenever the set of conditions for an event is satisfied, the corresponding event is triggered. Each activity 210 has one or more events 220 associated with it. Additional software stored in the computer system, which, for example, may be written in Power Builder, COBOL, or C programming languages, chooses an appropriate one of the events 220 for execution. Three events that may be executed during the course of the performance of an activity are, for example, "add," "delete" and "change." As illustrated in FIG. 5, three events identified when a user has selected the class registration activity may be "add class," "delete class," or "change class." Each of these events may be chosen for execution in response to a corresponding action by the user.

Referring back to FIG. 2, when an event 220 is chosen, a stored procedure makes a determination of all possible next steps 230 which are associated with that event 220. After the event 220 determines which next steps 230 are associated with it, the event makes a further determination as to which next steps 230 are to be chosen. This determination assures that a next user or group of users is able to perform a next activity 250, also referred to as a task 250, associated with the information entered by the user and/or data input from an automated process, such as an MRP system.

As shown in FIG. 6, a next step 230 may include the following information: (1) the next activity/task 250 to be performed; (2) the user/group of users responsible for performing the next activity/task 250; and (3) a message revealing to the user/group of users the nature of the next activity/task 250 to be acted upon. The next step 230 may also contain information or data disclosing the name of the entity within the organization in which the user/group of users fall under, which is ultimately responsible for performing the next activity/task 250. The list of information shown in FIG. 6, as described, is merely exemplary of the types of information that may be included in the next step 230.

Referring back to FIG. 2, based on the information contained in the chosen next steps 230, the computer system sends a message, representative of an associated category of next activity/task 250, to the To Do List 240 of the user or the group of users responsible for performing the next activity/task 250. Once the message is added to the To Do List 240 for the user or the group of users, the next activity/task 250 may be selected, viewed, and acted upon by the user associated with the To Do List 240 in a similar fashion as described for selecting the initial activity 210 which started the flow of the current information.

As shown in FIG. 7, a user may select a next activity/task 250 (see FIG. 2) from one or more To Do lists 240 located in the user's To Do List folder 330. In a preferred embodiment, a user may select a message 750 from a user's To Do List 700 personalized for the user. Alternatively, he or she may select a message 750 from the work group's To Do List 800 (see FIG. 8). In this example, the user selects the "New To Do List" 335 from the To Do Lists Folder 330.

To illustrate, the "New To Do List" 335 represents the user's personalized To Do List 700 set up for the user. Types of next activities/tasks 250 represented as messages 750 available to this user in his or her "New To Do List" To Do List window 700 are "Approve class registrations", "Registration confirmation," and "Select payment type for class (English)". These messages 750 represent next activities/tasks 250 categories subsequent to the initial activity 210 of class registration. In a preferred embodiment, the To Do list window 700 also displays the number of done/completed 760 and new 770 next activities/tasks 250 to the left of each message 750. In this example, the To Do list window 700, to the left of the "Approve class registrations" message 750, reveals that there are eight next activities/tasks 250 for this category, where six are new 770 and two are done 760.

Referring to FIG. 8, the user may also select a next activity/task 250 from a work group To Do List 700. This work group To Do List may be used when it does not matter which user among a group of users completes the next activity/task 250. For this example, the user has selected the Class Registration work group To Do List 336 from the To Do Lists folder 330. The message 750 revealed in the Class Registration work group To Do List is "Select payment type for class." This message 750 represents a next activity/task 250 subsequent to a prior user registering for a class. The user has four new 770 next activity/tasks 250 associated with this message 750.

Once the user selects the next activity/task 250 from the user's personalized To Do List 700 (see FIG. 7) or the user's work group To Do List 700 (see FIG. 8), referring to FIG. 2, the process for the flow of information which occurred after the initial activity 210 was selected may be repeated. As described above, the process would include the user entering relevant information into the activity window 400 (see FIG. 4) for the next activity/task 250, the user triggering an event 220, one or more next steps 230 being determined, and a corresponding message 750 (see FIG. 7) being added to the To Do List 240 of the user or work group responsible for performing the next activity/task 250. This cycle, inclusive of the user accessing the next activity/task 250 from the relevant To Do List 240, may continue until each piece of information is pushed entirely through the organization's predefined information flow path.

An example of the information flow process of the present invention is illustrated in FIG. 9. This example shows how the information flow process of the present invention is implemented, where the activity to be performed is to add a new part to a system for controlling a manufacturing operation.

In this example, the user (e.g., an engineer) chooses the "Part" activity 210 from his or her activity list (not shown) in order to create a new part for a manufacturing process within the organization. In response to this choice, the system displays a "Part" activity screen on the engineer's terminal screen 122 within activity window 400. This window 400 includes a prompt area 450, in which the engineer enters the number of the part (i.e., "06536").

When the engineer appropriately signals the system that the activity 210 is complete, the system responds by triggering execution of one of the events 220 associated with the activity 210. In this example, the event 220 triggered is the "Create a new Part" event 220.

The software "create a new part" event 220 then makes decisions based upon the addition of the part number to determine the next steps 230 to be undertaken in this flow process. For this example, the next steps 230 are "Review Part Planning information" to be done by the manufacturing manager and "approve part planning" (not shown) to be done by the quality department manager.

The "review part planning information" message 750, representative of a next activity/task 250 category, is then displayed by the system in the manufacturing manager's To Do List 240. In this example, the manufacturing manager has two messages 750, "Review part planning info" and "Define Part Eng. info.", listed in his personalized ("Things to do") To Do List window 700. The manufacturing manager may then select the "Review Part Planning info." message from his To Do List window 700, and the next activity/task 250 associated with this message is displayed in the manager's next activity/task window 400. Finally, this process for the flow of information through the organization is repeated as the manager enters relevant information into the window 400 for the next activity/task 250. The present invention then triggers an event 220 for determining the next steps 230 in the organizations procedure, and a message 750 representative of a next activity/task 250 category is added to another user's or work group's To Do List 240. As discussed above, this cycle continues until each piece of information is routed entirely through the organization's predefined information flow path (i.e., appropriately acted upon by all appropriate personnel in the organization in the proper sequence).

Set forth below is a description of the computer software for implementing a presently preferred embodiment of the present invention. As one of ordinary skill in the art would understand and appreciate, the following is merely one way of implementing the invention and many equivalent ways exist to achieve the same functions and results of the invention.

Referring to FIG. 1A, application programs are created using PowerBuilder code (application development software available from Powersoft, of Burlington, Massachusetts) and are stored on the client side 120 of the client/server network 100. Tables and stored procedures are created using SQL (Structured Query Language) code and are stored on the server side 110. Though PowerBuilder and SQL are the preferred software tools for the present invention, one of ordinary skill in the art would appreciate that the present invention could be implemented with many other equivalent types of software and/or development tools.

In a preferred embodiment, the PowerBuilder software is the preferred tool for creating the main application program of the present invention and the specific application programs to execute the windows 400 (see FIG. 4) for each activity 210 and next activity/task 250 (see FIG. 2). On the other hand, the SQL software is used primarily to create tables and stored procedures which interact with the tables and the PowerBuilder application programs to send information (e.g., data) back and forth on the network 100 (see FIG. 1A). In a preferred embodiment, the stored procedures are compiled and then interpreted by SQL engines.

Referring to FIG. 10, data may be stored in a table 1000, also known as a database structure, in rows 1010, also referred to as records, and columns 1020, also referred to as fields. Examples of tables utilized by the present invention and the relationships of these tables with one another are illustrated by FIGS. 11, 12, 13, 14 and 15. In yet another embodiment, the tables that are utilized

by the present invention and the relationships of these tables with one another are illustrated in FIGS. 14A, 14B, 14C, 14D, 14E, 14F and 14G, whereby like-named tables from FIGS. 11, 12, 13, 14 and 15 share a similar function as those depicted in FIGS. 14A, 14B, 14C, 14D, 14E, 14F and 14G.

The COLUMN(underscore)MASTER table 1330 (see FIG. 13) is used for mapping most of the columns in the tables by having certain information related to the columns in each table stored in this table. In a preferred embodiment, information on the column includes the column identifier (COL(underscore)ID), which is a numerical value representative of the position for that column in the COLUMN(underscore)MASTER table 1330; the column name (COL(underscore)NAME), which is the name the column is referred to in the other tables; the column label (COL(underscore)LABEL), which is the label used for viewing in the various windows; the column type (COL(underscore)TYPE), such as database datatype (i.e., character, integer, datetime, etc.); the column length (COL(underscore)LENGTH), which represents the size of the column; and the event column number (EVENT(underscore)COL(underscore)NBR), which is used to correlate an event identifier (EVENT(underscore)ID) in the EVENT(underscore)MASTER table 1220 with its particular column identifiers (COL(underscore)IDs).

FIG. 15 shows some of the primary tables 1000 used in a preferred embodiment of the present invention. Each line between two tables indicates that information (e.g., data) between these tables 1000 is shared in a one-to-many relationship. The head of the arrow (black dots in FIGS. 11-14) at one end of each line positioned next to one of the tables 1000 indicates that the table may have many rows (records) which are associated with one row (record) of the table 1000 positioned at the other end of the line.

Set forth below is a description of how, in a preferred embodiment, the applications programs, stored procedures, and tables 1000 interact to implement the information flow process of the present invention.

The software of the present invention is activated to begin executing when a user logs onto the computer system (e.g., client/server network) 100 from his or her end user workstation (e.g., an IBM compatible PC) 120 as described for FIG. 1A. After the user logs onto the system, the screen 122 on the user's terminal 126, as shown in FIG. 3, reveals the Browser mode window 310 which is preferably created with PowerBuilder software.

Referring to FIG. 16A, a user may create an activity list for his or her Activity Lists folder 320 by accessing the File mode and choosing the New (Ctrl + N) selection from the File list window 380. As shown in FIG. 16B, a "New Browser Objects" window 381 is displayed by the system in response to the above action by the user. The user then selects the "Activity List" as the new object type and enters the new activity list name (e.g., "Activities I do a lot") in the bottom left hand corner of the window. Finally, the user chooses which drawer (e.g., "DBS Home Drawer") and folder (e.g., "Activity List Folder") in which to put the activity list. The user also has the option to put the activity list in a drawer without putting the activity list into a folder.

In response to the above user inputs, the main application program sends the user's drawer number (referred to as the DRAWERNO in the tables), the folder number (FOLDERNO) assigned to the activity list, the description (DESCR) of the list (e.g., Activity List), and the location of the list (PARENTFOLDERNO) (e.g., the number assigned to the Activity List folder) from the client to a stored procedure at the server. In one embodiment, the name of the stored procedure is FOLDER\$insert(underscore)1. The stored procedure then stores this information in the WIJ(underscore)FOLDER table 1255 (see FIG. 12) and sends a message to the main application program that the information has been stored.

Each activity list is identified and stored under a corresponding folder number (FOLDERNO). The WIJ(underscore)FOLDERS table 1255 (see FIG. 12) is used to keep track of all FOLDERNOs. The CONTAINSIND column in the WIJ(underscore)FOLDERS table 1255 indicates whether the FOLDERNO refers to a folder or to a list. More details on the WIJ(underscore)FOLDERS table 1255 are provided below.

Referring to FIG. 16C, the user may add activities to an activity list by accessing, the Browser mode. The Browser mode window 310 reveals all folders 1670 and the activity lists 1680 or To Do Lists (not shown) contained within the folders 1670 for a particular drawer 305 of the user. The user may then select an activity list 1680 in which to add activities.

If an activity list 1600 is empty, as is the "Activities I do a lot" list 1680, then the activity list representation will be blank. However, if the activity list 1680 contains one or more activities or next activities/tasks, then the activity list representation will contain horizontal lines.

A user may add activities to an activity list by selecting, for example, the "Activities I do a lot" list 1680 in which to add one or more activities. Next, referring to FIG. 16D, the user selects the Browser list window 382 and chooses the Customize Activity List command from the Browser list window 382.

Referring to FIG. 16E, in response to the above actions of the user, the main application program displays a list of activities that the user may access in a Customize Activity List window 383. (The access is preferably based on security privileges predefined by an administrator of the system which is discussed further below). In determining which activities the user may access, the main application program sends the user's identifier (USER(underscore)ID) to a stored procedure. In one embodiment, the name of the stored procedure is psp(underscore)sel(underscore)avail(underscore)user(underscore)acts(underscore)1. This stored procedure then accesses the USER(underscore)SECURITY table 1125 (see FIG. 11) to determine which user specific activities, represented as activity identifiers (ACTIVITY(underscore)Ids), the user has security privileges.

Next, the stored procedure accesses the USER(underscore)MASTER table 1110 (see FIG. 11) to obtain group security identifiers (SEC(underscore)GROUP(underscore)Ids) associated with the user's USER(underscore)ID. The stored procedure then accesses the GROUP(underscore)SECURITY table 1155 (see FIG. 11) to determine which activities (ACTIVITY(underscore)Ids) the user, as a member of a group of users (work group), may access. Next, the stored procedure accesses the ACTIVITY(underscore)MASTER table 1210

(see FIG. 12) and uses the ACTIVITY(underscore)IDs to obtain a description of each activity (ACTIVITY(underscore)DESC) that the user may access. Finally, the stored procedure sends the ACTIVITY(underscore)IDs and each activity's corresponding activity description (ACTIVITY(underscore)DESC) back to the main application program at the client.

The main application program then displays each of the activities 1690 in the Customized Activity List window 383. The user may then decide which activities he or she will choose to put in an activity list by selecting specific activities from the display. In this way, a list may be created, where the list may include user specific and/or work group specific activities.

After the user has selected the activities to store in an activity list, such as "YOUR ACTIVITIES" list, the main application program sends the USER(underscore)ID, FOLDERNO, ACTIVITY(underscore)IDs and any sequencing of activities for the list (SEQ(underscore)NBR) to a stored procedure at the server. In one embodiment, the name of the stored procedure is psp(underscore)ins(underscore)usal(underscore)1. This stored procedure then accesses the USER(underscore)ACT(underscore)LIST table 1135 (see FIG. 11), where the stored procedure creates a row for each ACTIVITY(underscore)ID. The USER(underscore)ID and FOLDERNO are both stored in each row created for each ACTIVITY(underscore)ID. Moreover, the SEQ(underscore)NBR, if any, for each activity is also stored in the row for each ACTIVITY(underscore)ID. In a preferred embodiment of the invention, if no sequencing information is selected for the activity by the user, then all non-sequence specific activities will be listed in alphabetical order in the activity list.

After the stored procedure stores the activity information for a particular activity list in the USER(underscore)ACT(underscore)LIST table 1135, the stored procedure sends a message to the main application program at the client to this effect. The main application program then waits for the user to decide which action he or she wants to perform next.

A user may also create a To Do List for his or her To Do List folder 330 (see FIG. 3) by accessing, in a preferred embodiment, the File mode and choosing the New (Ctrl + N) selection from the File List window 380 (see FIG. 16A). As shown in FIG. 16F, the user then selects the "To Do List" as the object type and enters the new To Do List name (e.g., "My Urgent To Do Tasks") in the bottom left hand corner of the "New Browser Objects" window 381. Finally, the user chooses which drawer (e.g., "DBS Home Drawer") and folder (e.g., "To Do Folder") to put the To Do List. The main application program then sends the user's drawer number (DRAWERNO), the folder number (FOLDERNO) assigned to the To Do List, the description (DESCR) of the list (e.g., the To Do List), and the location of the list (PARENTFOLDERNO) (e.g., the number assigned to the To Do List folder) to a stored procedure. In one embodiment, the name of the stored procedure is FOLDER\$insert(underscore)1. This stored procedure then stores this information in the WIJ(underscore)FOLDERS table 1255 (see FIG. 12) and sends a message to the main application program that the information has been stored.

Referring to FIG. 16G, the user may move next activities/tasks categories from one To Do List 1685 to another To Do List 1685 by accessing the Browser mode. The user may then select a To Do List 1685 in which to add a next activity/task category. Like the activity lists 1680 (see FIG.

16C), the representations for the To Do Lists are blank when empty and contain horizontal lines when they contain one or more next activities/tasks.

The user moves a next activity/task category from one To Do List 1685 to another To Do List 1685 by selecting a To Do List 1685 which contains one or more next activity/task categories. As shown in FIG. 16H, the user selects the "New To Do List" To Do List 1685 from the Browser mode list window 310. The main application program then accesses a stored procedure to determine all next activities/tasks currently stored in the "New To Do List" To Do List 1685. Details on how the main application program and the stored procedure work together to compile the list of currently stored next activities/tasks is described below.

Next, the main application program displays these next activities/tasks organized by the message 750 representative of the next activity/task category in a Summary To Do Category window 384. The user then selects a next activity/task category to move to another To Do List 1685. For this example, the user selects the "You are a new member of a Workgroup" next activity/task category, which contains one uncompleted 770 next activity/task to be moved to another To Do List 1685.

Finally, as shown in FIG. 16I, the user selects the Browser list 382 and chooses the Move command from the Browser list 382. As shown in FIG. 16J, the main application program then reveals a list of possible To Do Lists 1685 in the Move mode list window 310 for the user to choose to move the next activity/task category. For this example, the user selects the "My urgent To Do Tasks" To Do List 1685. Therefore, as shown in FIG. 16K, when the user accesses the "My urgent To Do Tasks" To Do List in the Summary To Do List window 384, the "You are a new member of a Workgroup" message 750 next activity/task category with its one uncompleted next activity/task is revealed.

Referring to FIG. 16L, the user may also move individual next activities/tasks from one To Do List 1685 to another To Do List 1685. The user accomplishes this by selecting a To Do List containing next activities/task while in the Browser mode (not shown). The user then accesses the Detail To Do Category window 385 and a list of each individual next activity/task is revealed for a particular next activity/task category within the To Do List.

For this example, the user selects the "New To Do List" To Do List and details on two next activities/tasks are revealed for the "Select payment type for class" next activity/task category. The user then chooses a next activity task which, for this example, is ET201, to move to another To Do List. Next, referring to FIG. 16M, the user selects the Browser list 382 and chooses the Move command from the Browser list 382. As shown in FIG. 16N, the main application program then reveals a list of possible To Do Lists 1685 in the Browser mode list window 310 for the user to choose to move the next activity/task. For this example, the user selects the "My urgent To Do Task" To Do List 1685.

Therefore, as shown in FIG. 16O, when the user access the "New To Do List" in the Detail To Do Category window 385, the ET201 class payment next activity/task no longer exists. Moreover, as shown in FIG. 16P, when the user accesses the Summary To Do Category window 384 for the "My urgent To Do Tasks," the "Select payment type for class" next activity/task



category is revealed. The window 384 discloses that the "Select payment type for class" next activity/task contains one uncompleted next activity/task, which is the class payment for the ET201 class.

In another aspect of the invention, the user may create a customized folder in which to store activity lists. For example, a user may create a Class Registration folder for storing different lists of activities pertaining to registering for classes. The user accesses the File mode and chooses the New (Ctrl+N) selection from the File list window 380 (see FIG. 16A). The "New Browser Objects" window is then displayed. (see FIG. 16B). The user then selects the "Folder" as the new object type and enters the new folder name in the bottom left hand corner of the window. Finally, the user chooses the drawer in which to put the folder.

The following is an illustrative example of how the application programs at the client side and the stored procedures and tables at the server side interact to facilitate the flow of information in a work flow environment. For this example, a user registers for two classes.

In order to register for the classes, the user logs onto the system. In a preferred embodiment, after the user logs on, he or she selects the Browser mode from a list of possible modes displayed across the screen (not shown). Upon selecting the Browser mode, the main application program sends the user's USER(underscore)ID to a stored procedure at the server. The stored procedure then accesses the WIJ(underscore)DRAWERS table 1245 (see FIG. 12) to obtain information on the drawer number (DRAWERNO) and the description (DESCR) of the user's drawer.

The stored procedure then accesses the WIJ(underscore)FOLDERS table 1255 (see FIG. 12) to obtain information on all folders and lists, which are stored as folder numbers (FOLDERNOs), associated with the DRAWERNO. This information includes a description (DESCR) of the folder or list; a list indicator (CONTAINSIND), which reveals whether the FOLDERNO corresponds to a folder, an Activity List, a To Do List or some other list used in the system; and priority information (PARENTFOLDERNO), which is used to determine which folder each activity list and To Do List belongs in.

The main application program, as shown in FIG. 16C, uses the information sent to it by the stored procedure to display the folders 1670 and lists 1680 for the user's drawer 305 in the Browser mode list window 310. As shown in FIG. 17A, to illustrate, the user's drawer 305 is described as DBS Home Drawer in the Browser mode list window 310. Moreover, the user's Drawer 305 contains an Activity Lists folder 320, a To Do Lists folder 330 and a Mail folder 1720. The Activity Lists folder 320 contains a "Sample Class Registration" list 1710, as well as other lists such as a "Management Reporter" list 1711 and a "Product Support" list 1712. The To Do Lists folder 330, for this example, contains only the user's personalized To Do List, referred to as the "New To Do List" 335.

For this example, to register for classes, the user selects the "Sample Class Registration" list 1710 and, in response, the system displays the user's customized "Sample Class Registration" list window 1750, with pre-selected activities.

The computer system of the present invention obtains the activities for the customized activity list window 1750 (e.g., "Sample Class Registration" list window) by having the main application program send the USER(underscore)ID and FOLDERNO for the activity list selected by the user to the stored procedure at the server. In one embodiment, the name of the stored procedure is psp(underscore)sel(underscore)user(underscore)cando(underscore)list(underscore)1. This stored procedure then accesses the USER(underscore)ACT(underscore)LIST table 1135 (see FIG. 11) to obtain each activity, via its ACTIVITY(underscore)ID, that is associated with the USER(underscore)ID and FOLDERNO, as well as any SEQ(underscore)NBR information pertaining to the prioritizing of these activities set up by the user for displaying each activity in the activity list window 1750. The stored procedure also accesses the ACTIVITY(underscore)MASTER table 1210 (see FIG. 12) to obtain information on the type of activity (ACTIVITY(underscore)TYPE) (e.g., a PowerBuilder type window or an executable file type) and the command line (EXEC(underscore)NAME) for the activity window application program (e.g., Class Registration activity window) or executable application program which is used to execute the activity. The stored procedure then returns this information to the main application program so that the activity list window 1750 may be displayed with the activities listed in sequence specific or alphabetical order where no sequence specific information for an activity is indicated.

For this example, the user has selected the Class Registration, Class Payment, Registration Approval, and Activity activities 1760 for his or her "Sample Class Registration" list 1750. Moreover, the user has chosen to sequentially list the activities such that the Class Registration Activity has the highest SEQ(underscore)NBR, with Class Payment and Registration Approval having lower SEQ(underscore)NBRs and Activity having the lowest or no SEQ(underscore)NBR.

From the "Sample Class Registration" list window 1750, the user then selects the Class Registration activity from the list of activities in order to register for a class. As shown in FIG. 17B, the main application program then calls the activity window application program represented by its EXEC(underscore)NAME by issuing an "open" command (a PowerBuilder command). The main application program then reveals the Class Registration activity in a Class Registration activity window 1700. If the user had selected to run an executable file, then the main application program would have called the executable file represented under its EXEC(underscore)NAME by sending it a "run" command (a PowerBuilder command).

The computer system reveals the activity window 1700 by having the main application program access a Class Registration application program responsible for the Class Registration activity. The Class Registration application program contains information on the structure of the window and the headings 420 (e.g., Student, Class, Class Description, and Credit Status for the Class Registration activity). The Class Registration application program then executes a stored procedure. In one embodiment, the name of the stored procedure is psp(underscore)sel(underscore)sam1(underscore)1. The server then requests this stored procedure to send it back information on available classes for the user. The stored procedure then accesses the SAMPLE(underscore)CLASS table 1360 (See FIG. 13) to obtain information on each class (CLASS) and a description (DESCRIPTION) of the class. The stored procedure then

sends this information, also referred to as values, back to the Class Registration application program.

After the Class Registration application program receives the headings 420 and their associated values 430, this information is displayed in the Class Registration activity window 1700. For this example, one class at a time is listed as a value 430 next to the Class heading 420 with the corresponding information for the class filling portions of the rest of the activity window 1700. The user may then scroll (e.g. with the arrow keys on the keyboard) through all possible classes that the user has access to register.

After the user has selected a class in which to register (e.g., ET201 - Ethics in the Workplace) along with the credit status (e.g., graduate), the user saves the information. The user may save the information by clicking over the "save file" icon, represented as a floppy disk (not shown), with the mouse or simply pressing the Control and S keys simultaneously on the keyboard.

The Class Registration application program saves the Class Registration activity information (e.g., ET201 for class and graduate for credit status) in the following fashion. As shown in FIG. 17D, at step 1790, the Class Registration application program (written in PowerBuilder) calls a stored procedure, at step 1791, to format the information for certain columns in the associated table at the server. For this example, the information is formatted for the class, student, and credit columns in the SAMPLE(underscore)REG table 1365 (see FIG. 13). At step 1792, the formatted information is returned to the Class Registration application program, which, at step 1793, then sends the formatted information to a stored procedure at the server. In one embodiment, the name of this stored procedure is psp(underscore)ins(underscore)sam2(underscore)1. This stored procedure then stores each piece of information (each value) in its corresponding class, student and credit column in the SAMPLE(underscore)REG table 1365. Finally, at step 1794, the stored procedure returns a message to the Class Registration application program that indicates that the information has been saved.

At step 1795, the Class Registration application program then determines whether the information was successfully saved. If not, the application program proceeds to step 1796 and returns control to the user. However, if the information was saved successfully, then the Class Registration application program determines which event to trigger. For this example, possible events associated with the Class Registration activity are "Add Class" and "Change Class". To illustrate, the user has added a new class. Therefore, the "Add Class" event is selected, along with the corresponding stored procedure for this event. In one embodiment, the name of the "Add Class" stored procedure is pamsam2ins. At step 1797, the Class Registration application program then executes the Trigger Event function stored procedure to determine the next activities/tasks and users/workgroups responsible for completing the next activities/tasks associated with the Class Registration activity. In one embodiment, the name of the Trigger Event stored procedure is called pam0011(underscore)trig(underscore)jam(underscore)event. In executing the Trigger Event stored procedure, the Class Registration application program sends information to the Trigger Event stored procedure at the server.

The information sent to the Trigger Event stored procedure includes an event identifier (EVENT(underscore)ID) for the corresponding stored procedure (e.g., pamsam2ins), the entity

value (NEXT(underscore)STEP(underscore)ENT(underscore)VAL) (e.g., plant, site, organization, etc.) responsible for performing the next activity/task, the USER(underscore)ID for the user who completed the activity, the ACTIVITY(underscore)ID for the activity just completed (e.g., Class Registration), and the priority of the subsequent next activity/task (MSG(underscore)PRIORITY) set by the user who has just completed the activity. As shown in FIG. 17C, the user who just completed the activity may set the priority of the subsequent next activity/task by selecting the Set Task Priority option in the Options Mode. Other information may include whether a user or a work group (OWNER(underscore)TYPE) is responsible for the next activity/task and the identification for the user or work-group (USER(underscore)ID or MSG(underscore)GROUP(underscore)ID).

Referring to FIG. 18, a flow diagram is provided, for a preferred embodiment of the Trigger Event function stored procedure. This flow chart illustrates a process for determining the next activity/task(s), user/work group responsible for performing the next activity/task(s), and the like, and creating the next activity/task(s).

First, at step 1810, the Trigger Event function stored procedure determines whether the event (e.g., pamsam2ins for the "Add Class" event) is enabled. This is accomplished by accessing the EVENT(underscore)MASTER table 1220 (see FIG. 12) to determine if the ENABLED column for the EVENT(underscore)ID (e.g., pamsam2ins) contains a one or a zero. In a preferred embodiment, if the ENABLED column contains a zero, then the event is disabled, the Trigger Event function stored procedure is exited at step 1812, and control is returned to the Class Registration application program.

However, if the ENABLED column contains a one, then the event is enabled and the Trigger Event function stored procedure proceeds to step 1815. At step 1815, column identifiers (COL(underscore)IDs) are obtained from the EVENT(underscore)MASTER table 1220 for the corresponding EVENT(underscore)ID. Thus, the Trigger Event function stored procedure is able to determine which COL(underscore)IDs for the given event will be used to pass the values pertaining to the event. For this example, possible COL(underscore)IDs for the pamsam2ins EVENT(underscore)ID could be 1060 for COL(underscore)ID(underscore)1 representative of the Class heading, 1061 for COL(underscore)ID(underscore)2 representative of the Student heading, and 1062 for COL(underscore)ID(underscore)3 representative of the Credit heading.

The Trigger Event function stored procedure then proceeds to step 1820 to obtain the first of all possible next steps which are associated with the event. In doing so, at step 1823, the Trigger Event function stored procedure accesses the Next Step table 1225 (see FIG. 12) to determine if any next steps exists for the EVENT(underscore)ID. This is accomplished by determining if any rows (records) exists for the particular EVENT(underscore)ID in the Next Step table 1225. If there are no rows for the EVENT(underscore)ID in the Next Step table 1225, then the Trigger Event function stored procedure is exited at step 1826.

However, if rows for the EVENT(underscore)ID do exist in the Next Step table 1225, then, at step 1830, the Trigger Event function stored procedure determines whether the first row, which represents a particular next activity/task (ACTIVITY(underscore)ID), is enabled (ENABLED). If the ENABLED column is disabled (e.g., zero), then the Trigger Event function stored procedure

returns to step 1820 to obtain the next step, if there are any. This loop continues until an enabled next step for a next activity/task corresponding to the EVENT(underscore)ID is encountered. If there are no rows enabled for the EVENT(underscore)ID, then, after the last one is encountered, the Trigger Event stored procedure will return to the Class Registration application program.

On the other hand, if a first row encountered for an EVENT(underscore)ID is enabled (e.g., one), then the Trigger Event function stored procedure obtains the message identifier (MSG(underscore)ID) representing the next activity/task category and the next activity/task identifier (ACTIVITY(underscore)ID) and proceeds to step 1835. At step 1835, the Trigger Event function stored procedure accesses the Next Step Options table 1230 (see FIG. 12) for the entity (NEXT(underscore)STEP(underscore)ENT(underscore)VAL) responsible for the next activity/task, and the stored procedure proceeds to step 1838

At step 1838, if rows exist for the NEXT(underscore)STEP(underscore)ENT(underscore)VAL, then the Trigger Event stored procedure proceeds to step 1850. However, if no rows exist for the NEXT(underscore)STEP(underscore)ENT(underscore)VAL, then the Trigger Event stored procedure proceeds to step 1840 to obtain any default values sent from the Class Registration application program which define the entity responsible for the next activity/task. This information is stored in the Next Step Options table 1230 in the NEXT(underscore)STEP(underscore)ENT(underscore)VAL column (see FIG. 12). In one embodiment, specific entities may be delineated with an identifier or an asterisk (\*) may be used to indicate that every user (enterprise wide) may have access to the next activity/task. The Trigger Event function stored procedure then proceeds to step 1843 to determine if any default values are available. If no default values are available, then the Trigger Event function stored procedure returns to step 1820 to determine if there are any other next steps to be evaluated and acted upon as described above. However, at step 1843, if default values do exist, then the Trigger Event stored procedure proceeds to step 1850.

At step 1850, the Trigger Event stored procedure accesses the Next Step Options table 1230 to determine the user (USER(underscore)ID) or work group (MSG(underscore)GROUP(underscore)ID) responsible for the next activity/task. In doing so, it checks the IGNORE(underscore)OVERRIDE column for the row to determine whether the USER(underscore)ID or MSG(underscore)GROUP(underscore)ID values should be used. If the IGNORE(underscore)OVERRIDE is enabled, then the values identified in the column are used. However, if the IGNORE(underscore)OVERRIDE is disabled, then the values sent from the Class Registration application program are used. On the other hand, if the Class Registration application program did not send any values, then the values from the Next Step Options table 1230 are used. Next, the Trigger Event function stored procedure proceeds to step 1861, where it is determined whether the user is on the present server. If not, then at step 1862, asynchronous RPC is sent to the remote server's messages queue.

Finally, the Trigger Event stored procedure proceeds to step 1870 where information pertaining to the next activity/task is added to the MESSAGE(underscore)QUEUE table 1140 (see FIG. 11) by creating a row for the next activity/task. The information in the row may then be used later for the responsible user's or work group's To Do List. This information includes the OWNER(underscore)ID which is the USER(underscore)ID;

MSG(underscore)GROUP(underscore)ID (for a work group) or override value; the OWNER(underscore)TYPE, which indicates whether the OWNER(underscore)ID belongs to a user or a work group; the ACTIVITY(underscore)ID for the next activity/task; the MSG(underscore)ID pertaining to the next activity/task category; and CREATE(underscore)TIME, which indicates the date and time the next activity/task was created.

Other information that may be stored in the new next activity/task row includes the MSG(underscore)SEQ(underscore)NBR, which indicates the priority associated with the message representative of the new activity/task category, where the priority is assigned to the next activity/task category by the user or work group to perform the next activity/task in the Detail To Do Category window (see FIG. 16L); the FOLDERNO, which indicates the user's or work group users' list associated with the next activity/task; and the NEXT(underscore)STEP(underscore)ENT(underscore)VAL, which indicates the plant, site, organization or the like ultimately responsible for the next activity/task.

Finally, particular information relating to the class in which the user has registered is stored in a COL(underscore)VAL associated with a specific COL(underscore)ID. For example, COL(underscore)ID(underscore)1 which is 1060 and represents the Class heading correlates with COL(underscore)VAL(underscore)1 which stores the class value, ET201; COL(underscore)ID(underscore)2 which is 1061 and represents the student heading correlates with COL(underscore)VAL(underscore)2 which stores the student value, DBS; and COL(underscore)ID(underscore)3 which is 1062 and represents the credit heading correlates with COL(underscore)VAL(underscore)3 which stores the credit value, graduate.

After the Trigger Event stored procedure has completed step 1870 by adding the pertinent information to the new next activity/task row of the MESSAGE-QUEUE table 1140, then the stored procedure returns to step 1820 to determine if there are any other next steps to be evaluated and acted upon as described above. After the Trigger Event stored procedure has acted upon all the next steps, as described above, the stored procedure proceeds from step 1820 to step 1826. At step 1826, the Trigger Event stored procedure is exited and control is returned to the Class Registration application program.

The Class Registration application program then displays a blank Class Registration activity window (not shown) and waits for the user to either register for another class or exit the Class Registration activity. For this example, as shown in FIG. 19, the user registers for a second class, SC101 (Security Administration) with an audit credit status. Therefore, the user executes the save command to save the class. Next, the procedure described above is repeated as the Class Registration application program formats the information entered by the user, and sends this information to be saved by the stored procedure. In one embodiment, the name of the stored procedures is psp(underscore)ins(underscore)saml(underscore)1. After the information has been stored, the Class Registration application program determines which event to trigger and triggers the event, via the Trigger Event stored procedure, to determine the next activities/tasks and users/work groups responsible for completing the next activities/tasks.

After the Class Registration application program receives a message from the Trigger Event stored procedure that all next activities/tasks have been added to users/work group's To Do List, the Class Registration application program again waits for the user to either register for another class or exit the Class Registration activity. For this example, as shown in Fig. 20, the user decides to leave the Class Registration activity window and return to the Browser mode list window 310. In a preferred embodiment, the user may press the F11 key to return the user to the Browser mode list window 310.

From the Browser mode list window 310, the user may then decide whether to access another list in the Activity List Folder 320 to begin another activity or to access the user's personalized To Do List ("New To Do List") 335 in the To Do Lists folder 330. In this example, the user selects the "New To Do List" 335, and a list of next activities/tasks is displayed in the Summary To Do Category window 384.

The computer system obtains the next activities/tasks for the Summary To Do Category window 384 (e.g., "New To Do List" window) by having the main application program send the USER(underscore)ID and FOLDERNO selected for the To Do List by the user to the stored procedure. In one embodiment, the name of the stored procedure is psp(underscore)sel(underscore)mque(underscore)detail. For this example, the FOLDERNO corresponds to the user's personalized To Do List (e.g., "New To Do List").

The stored procedure then accesses the MESSAGE(underscore)QUEUE table 1140 (see FIG. 11) to obtain the message via the message identifier (MSG(underscore)ID) for each next activity/task category corresponding to the user's USER(underscore)ID (identified as the OWNER(underscore)ID) and FOLDERNO. The stored procedure also obtains any priority information regarding the message and the next activities/tasks represented by the message from the MSG(underscore)SEQ(underscore)NBR and MSG(underscore)PRIORITY columns, respectively. The status of each individual next activity/task, as to whether it has previously been completed, is also obtained by the stored procedure from the MSG(underscore)STATUS column. Moreover, the stored procedure obtains the ACTIVITY(underscore)ID, COL(underscore)Ids, and COL(underscore)VALs associated with each MSG(underscore)ID for all its corresponding next activities/tasks. The stored procedure also accesses the ACTIVITY(underscore)MASTER table 1210 (see FIG. 12) to obtain information on the type of activity (ACTIVITY(underscore)TYPE) (e.g., PowerBuilder type window or executable file type) and the command line (EXEC(underscore)NAME) for the activity window application program or executable application program which is used to execute the next activity/task. The stored procedure then sends this information back to the main application program, which organizes the messages based on priority, calculates the number of completed and uncompleted next activities/tasks for each message, and reveals the messages 750 and completed (done) 760 and uncompleted (new) 770 information in the Summary To Do Category window 384.

For the "New To Do List" Summary To Do Category window 384, there are five messages 750. These messages 750 are "Activity Category(s) have been updated", "Category Added, Check Configuration", "Select payment type for class", "You are a new member of a Workgroup", and "You are now a Security Administrator". To the left of each message 2080 is the "Done" column 760 and the "New" column 770, which represent the number of next activities/tasks completed

and uncompleted for the particular message 750. To illustrate, the "Category Added, Check Configuration," message 750 has two Done (completed) 760 and nine New (uncompleted) 770 next activities/tasks associated with the message 750.

As shown in FIG. 21, the user may also access a Detailed To Do Category window 385. In this window 385, when the user accesses the To Do List, one message 750 is revealed at a time. For this example, the main application program is displaying the "Select payment type for class" message 750. The main application program reveals, to the left of the message 750, the number of next activities/tasks 2165 that exist, and discloses, to the right of the message 750, the next activity/task 250 associated with this message 750. The main application program also lists, below the message 750, key information relating to each next activity/task 250. From left to right, this includes whether the next activity/task 250 has been completed 2170, the priority 2130 associated with each next activity/task 250, and, for this example, the class heading 2140 and student headings 2120.

To illustrate, the number of existing next activities/tasks 250 associated with the "Select payment type for class" message 750 are two, and the next activity/task 250 associated with this message 750 is Class Payment. Further, the next activities/tasks 250 relate to two classes 2140, ET201 and S2101, sought to be registered by a user/student 2120, DBS. For this example, DBS, the one originally registered for both of these classes, is also responsible for selecting a payment type for each class. The window 385 also reveals that both of these next activities/tasks 250 are uncompleted 2170 and have medium priority status 2130.

The user may select a category for a next activity/task to act upon from either the Summary To Do Category window 384 or the Detailed To Do Category window 385. For this example, the user selects the Class Payment next activity/task category, represented by the "Select payment type for class" message, from the Summary To Do Category window 384, as shown in FIG. 22A. The main application program then sends the FOLDERNO for the To Do List and the MSG(underscore)ID for the category selected to the next activity/task application program. For this example, the FOLDERNO and MSG(underscore)ID would be sent to the Class Payment application program represented by its EXEC(underscore)NAME. In one embodiment, the EXEC(underscore)NAME of the Class Payment application program is pam0510(underscore)payment.

The next activity/task application program then sends the FOLDERNO and MSG(underscore)ID to a stored procedure which determines the most prioritized next activity/task for the next activity/task category. In one embodiment, the name of the stored procedure is the Next Task stored procedure, which is discussed in further detail below and is illustrated in FIG. 26. After determining the most prioritized next activity/task, the Next Task stored procedure sends information on this next activity/task (class payment), represented by its ACTIVITY(underscore)ID, to the next activity/task application program or executable application program.

In this example, the class payment next activity/task category only contains one next activity/task. Therefore, the Next Task stored procedure selects the next activity/task, and the Next Task stored procedure send the Class Payment activity application program the next



activity/task. The Next Task stored procedure also sends the Class Payment application program other information regarding this next activity/task. As shown in FIG. 22B, the Class Payment application program then reveals the Class Payment next activity/task in a Class Payment activity window 2400, as described above, for the Class Registration activity.

The user may also access next activities/tasks via, the Options mode. To illustrate, if the same user is responsible for an activity and corresponding next activity/task or next activity/task and corresponding next activity/task that occur sequentially, then the user may access the next activity/task from the activity or next activity/task window via the Options mode. For example, since the user (DBS) is responsible for both the Class Payment next activity/task and the just completed Class Registration activity, then DBS may access the Class Payment next activity/task from the Class Registration window via the Options mode.

To illustrate, as shown in FIG. 22C, after DBS had completed registering for the SC101 class in the Class Registration activity window 1950, DBS accesses the Options mode. In doing so, the Class Registration application program calls the Options application program, which reveals a list of options 2250 for the user to select from. Next, in order to access the sequential next activity/task in connection with the Class Registration activity, the user selects the Next Step option 2260 from the Options mode list 2250.

Upon accessing the Next Step option 2260, the Option application program calls the Class Registration application program to obtain information to be sent to a Next Step application program. The information obtained is the USER(underscore)ID, ACTIVITY(underscore)ID, and a key value, which, for this example, is SC101 for the class value (COL(underscore)VAL(underscore)1) as well as its corresponding column identifier (COL(underscore)ID(underscore)2). The Options application program then calls the Next Step application program and sends the USER(underscore)ID formatted as the OWNER(underscore)ID and FROM(underscore)USER(underscore)ID for DBS, the ACTIVITY(underscore)ID formatted as the FROM(underscore)ACT(underscore)ID for the Class Registration activity, the COL(underscore)VAL(underscore)1 for SC101, and the COL(underscore)ID(underscore)1 for the class heading to the Next Step stored procedure.

The Next Step stored procedure is illustrated in FIG. 23. In one embodiment, the Next Step stored procedure is called `psp(underscore)mque(underscore)next(underscore)step(underscore)1`. At step 2310, the Next Step stored procedure obtains a list of all possible next steps in connection with the just completed activity for that user. In doing so, the Next Step stored procedure accesses the MESSAGE(underscore)QUEUE table 1140 (see FIG. 11) to obtain all next activities/tasks (represented as ACTIVITY(underscore)Ids) and their message identifiers (MSG(underscore)Ids) for the OWNER(underscore)ID and FROM(underscore)USER(underscore)ID corresponding to DBS, the FROM(underscore)ACT(underscore)ID for the Class Registration activity, and the key value corresponding to SC101 (COL(underscore)VAL(underscore)1) for the class heading (COL(underscore)ID(underscore)1). The Next Step stored procedure then proceeds to step 2320.

At step 2320, the Next Step stored procedure calculates the number of next steps encountered. If the number of next steps encountered equals zero, then the Next Step stored procedure is exited

at step 2315 and a message to this effect is sent back to the Next Step application program which reveals this to the user. If there is only one Next Step, then the stored procedure proceeds to step 2350. However, if the number of next steps equals one or more, then the Next Step stored procedure proceeds to step 2330.

At step 2330, the Next Step stored procedure sends the ACTIVITY(underscore)Ids and MSG(underscore)Ids to the Next Step application program, which lists each next activity/task in a window (not shown) that corresponds to the just completed Class Registration activity for the SC101 class. The user may then select which next activities/tasks he or she would like to act upon. The Next Step application program then sends the ACTIVITY(underscore)Ids for these next activities/tasks selected back to the Next Step stored procedure. The Next Step stored procedure then proceeds to step 2340.

At step 2340, the Next Step stored procedure calculates the number of next steps selected by the user. If the number of next steps selected equals zero, then the Next Step stored procedure is exited at step 2345 and a message to this effect is sent back to the Next Step application program which exits the Options window 1950 (see FIG. 22C) accordingly. However, if the number of next steps selected equals one or more, then the Next Step stored procedure proceeds to step 2350.

At step 2350, the Next Step stored procedure accesses the MESSAGE(underscore)QUEUE table 1140 (see FIG. 11) and obtains the ACTIVITY(underscore)ID, FOLDERN0, and information contained in the COL(underscore)Ids and COL(underscore)VALs for the first next activity/task selected. The Next Step stored procedure then proceeds to step 2360, where it determines whether there are any other next activities/tasks which need to be accessed from the MESSAGE(underscore)QUEUE table 1140. If not, then the Next Step stored procedure proceeds to step 2370.

However, if there are other next activities/tasks to be accessed from the MESSAGE(underscore)QUEUE table 1140, then the Next Step stored procedure returns to step 2350 and the above-mentioned information is obtained for the second next activity/task. This cycle continues until the pertinent information for each next activity/task selected by the user has been obtained by the Next Step stored procedure. The Next Step stored procedure then proceeds to step 2370.

At step 2370, the information related to each next activity/task selected by the user is sent back to the Next Step application program. This includes all ACTIVITY(underscore)Ids, FOLDERN0s, COL(underscore)Ids and COL(underscore)VALs. The Next Step stored procedure also sends a message as to the sequence that each next activity/task should be executed.

The Next Step application program then calls the application program responsible for performing the first next activity/task. If there are other next activities/tasks that need to be executed after the first (or current) next activity/task, then a message is sent to the application program indicating that it should call the Next Step application program upon completion of the first (current) next activity/task.

For this example, the Class Payment application program is called up. Therefore, referring to FIG. 24, the Class Payment application program reveals a Class Payment activity window 2400. As described in the foregoing, the user (DBS) may then complete the next activity/task by selecting the payment type (e.g., cash), and events and next steps will be triggered accordingly.

In another aspect of the present invention, the user may utilize the Options mode from a just completed activity or a just completed next activity/task window to simply call some next activity/task to the screen. The next activity/task chosen is selected, based on the following criteria. First, the highest priority next activity/task for the same next activity/task category. If no next activities/tasks exist for the preceding next activity/task category or the user has just completed an activity, then the highest priority next activity/task from the user's personalized To Do List is selected. If no next activities/tasks exist in the user's personalized To Do List, then the highest priority next activity/task contained in any other To Do List in which the user has access is selected.

To illustrate, as shown in FIG. 25, after a user has completed registering for the SC101 class in the Class Registration activity window, the user (DBS) accesses the Options mode. In doing so the Class Registration application program then calls up the Options application program, which reveals a list of options 2250 for the user to select from. Next, the user selects the Next Task option 2560 from the Options mode list 2250.

Upon accessing the Next Task option, the Option application program calls the Class Registration application program to obtain information to be sent to the Next Task application program. The information obtained is the USER(underscore)ID, OWNER(underscore)type (user or work group), and ACTIVITY(underscore)ID. If the Options mode had been accessed from an application program associated with a next activity/task, then the MSG(underscore)ID and the FOLDERNO for the To Do List associated with the next activity/task would also have been obtained. The Option application program then calls the Next Task application program and sends it the USER(underscore)ID formatted as an OWNER(underscore)ID, the OWNER(underscore)TYPE and the ACTIVITY(underscore)ID. The Next Task application program then sends this information to the Next Task stored procedure.

In one embodiment, the Options application program calls the pam0015(underscore)next(underscore)msg (Next Task) application program, which sends the OWNER(underscore)ID, OWNER(underscore)TYPE, and ACTIVITY(underscore)ID to the psp(underscore)sel(underscore)mque(underscore)next(underscore)msg(underscore)1 (Next Task) stored procedure. If the Options mode had been accessed from an application program associated with a next activity/task, then the MSG(underscore)ID and FOLDERNO corresponding to the next activity/task would also have been sent to the Next Task stored procedure.

Referring to FIG. 26, the Next Task stored procedure is illustrated. At step 2610, the Next Task stored procedure checks the OWNER(underscore)TYPE to determine if the user has accessed the Next Task option from a user or work group list. If the user has accessed the Next Task option from a user list, then the Next Task stored procedure proceeds to step 2625. However, if the user has accessed the Next Task option from a work group list, then the Next Task stored procedure proceeds to step 2615.

At step 2615, the Next Task stored procedure determines if the user is currently a manager or member of the work group, in case the user's access privileges have been revoked since the time the user accessed the current work group list. If the user no longer has access privileges, then the Next Task stored procedure is exited at step 2618 and a message to this effect is sent back to the Next Task application program which reveals this to the user. If the user still has access privileges to the work group list, then the Next Task stored procedure proceeds to step 2625.

At step 2625, the Next Task stored procedure determines whether a folder number (FOLDERNO) for a To Do List was sent from the Next Task application program. If so, then the Next Task stored procedure understands that the Next Task option was called from either an individual user's or work group user's To Do List, and the stored procedure proceeds to step 2650.

At step 2650, the Next Task stored procedure determines if the To Do List represented by the FOLDERNO contains any of the same next activities/tasks categories represented as the MSG(underscore)IDs from the just completed next activity/task. If the FOLDERNO for the To Do List sent contains at least one same next activity/task category, then the Next Task stored procedure proceeds to step 2660. However, if the To Do List does not contain at least one same next activity/task category, the stored procedure proceeds to step 2652.

At step 2652, the Next Task stored procedure determines if the FOLDERNO for the To Do List sent contains at least one next activity/task. If the To Do List sent contains at least one next activity/task, then the Next Task stored procedure proceeds to step 2655.

At step 2655, the stored procedure evaluates each next activity/task in the user's To Do List sent to determine which next activity/task category (MSG(underscore)ID) has the highest sequence number and priority next activity/task. The sequence number is assigned to a next activity/task category by the user who will act upon the next activity/task, and the priority is assigned to the next activity/task by the user who is responsible for initiating the next activity/task. The Next Task stored procedure then proceeds to step 2660. On the other hand, at step 2652, if the Next Task stored procedure determines that the FOLDERNO for the To Do List sent does not contain at least one next activity/task, then the stored procedure proceeds to step 2630.

At step 2625, if a folder number was not sent to the Next Task stored procedure, then the stored procedure interprets this to mean that the activity is not currently in the context of a To Do List. An example of this occurs when the user selects the Next Task option from one of the user's activity lists. In this case, the stored procedure proceeds to step 2630.

At step 2630, the Next Task stored procedure obtains the folder number for the user's personalized To Do List, represented as the TO DO(underscore)FOLDERNO, from the USER(underscore)MASTER table 1110 (see FIG 11). Next, the Next Task stored procedure proceeds to step 2633 to determine if there is at least one next activity/task in the user's personalized To Do List.

If there is at least one next activity/task in the user's personalized To Do List, then the Next Task stored procedure proceeds to step 2635 to determine which next activity/task category,

represented in the MSG(underscore)ID column, in the user's personalized To Do List has the highest sequence number and priority task. The Next Task stored procedure then proceeds to step 2660.

If the Next Task stored procedure does not find a next activity/task in the user's personalized To Do List at step 2633, then the stored procedure proceeds to step 2637. At step 2637, the Next Task stored procedure, scans all the user's other To Do Lists by FOLDERNO in the MESSAGE(underscore)QUEUE table 1140 (see FIG. 11) to determine if any other To Do Lists for the user contain next activities/tasks. If the Next Task stored procedure does not find a next activity/task in any of the user's other To Do Lists, then the stored procedure is exited at step 2645 and a message to this effect is sent back to the Next Task application program which reveals this to the user.

However, if one or more of the user's To Do Lists contain at least one next activity/task, then the Next Task stored procedure proceeds to step 2640. At step 2640, the Next Task stored procedure then evaluates each next activity/task in each of the user's To Do Lists to determine which next activity/task has the highest sequence number and priority task. The Next Task stored procedure then proceeds to step 2660.

At step 2660, the Next Task stored procedure selects a next activity/task based on the following hierarchy. First, the highest sequence number (MSG(underscore)SEQ(underscore)NBR) in the MESSAGE(underscore)QUEUE table 1140 for the task category, unless the stored procedure has proceeded from step 2650 such that the next activity/task category (MSG(underscore)ID) is already chosen. Second, the highest priority (MSG(underscore)PRIORITY) in the MESSAGE(underscore)QUEUE table 1140, where the priority may be high, medium, and low, for each next activity/task for the selected next activity task category. Third, the oldest create time (CREATE(underscore)TIME) in the MESSAGE(underscore)QUEUE table 1140 for the selected next activity/tasks with the highest priority in the selected next activity/task category.

If for some reason there are not any next activities/tasks encountered by the Next Task stored procedure in step 2660 (e.g., the user's access privilege to a To Do List was removed during the steps of the Next Task stored procedure as discussed above), then the stored procedure is exited at step 2665 and a message to this effect is sent back to the Next Task application program which reveals this to the user. However, if a next activity/task is selected at step 2660, then the Next Task stored procedure proceeds to step 2670. At step 2670, the Next Task stored procedures determines if the next activity/task selected is a work group next activity/task. If the next activity/task is not a work group next activity/task, then the stored procedure proceeds to step 2685.

However, at step 2670, if the next activity/task is a work group next activity/task, then the Next Task stored procedure proceeds to step 2675. At step 2675, the stored procedure determines whether the work group next activity/task still needs to be acted upon (e.g., has not already been assigned to another user within the work group or simply no longer needs to be acted upon). If the work group next activity/task still needs to be acted upon, then the stored procedure proceeds to step 2685. However, if the work group next activity/task no longer needs to be acted upon, the Next Task stored procedure proceeds to step 2680 where the stored procedure is instructed to

"Try Again" by proceeding to step 2625 in order to locate another next activity/task as discussed above.

At step 2685, the Next Task stored procedure updates the message status (MSG(underscore)STATUS) column for the next activity/task (ACTIVITY(underscore)ID) selected to a "viewed" status. This status indicates that user is about to view the next activity/task. This status is one of several possible statuses. Other possible statuses include the "new" status, which indicates the next activity has recently been created and there have not been any users notified of its existence; the "notified" status which indicates that at least one user has been notified that the next activity/task exists; and the "complete" status which indicates that the next activity/task has been viewed, acted upon, and completed by a user.

Next, the Next Task stored procedure proceeds to step 2690, where, as long as the next activity/task still needs to be acted upon, pertinent next activity/task values are obtained from the MESSAGE(underscore)QUEUE table 1140 (see FIG. 11) to be sent to the activity application program which will perform this next activity/task. However, if for some reason the next activity/task no longer needs to be acted upon for whatever reason, then the Next Task stored procedure proceeds to step 2680 where the stored procedure is instructed to "Try Again" by proceeding to step 2625 in order to locate another next activity/task as discussed above.

Finally, if the pertinent values are obtained by the Next Task stored procedure at step 2690, then the stored procedure is exited at step 2699. At step 2699, the pertinent values are then sent back to the Next Task application program, which forwards them to the appropriate activity application program to perform the next activity/task.

For this example, referring to FIG. 27, the next activity/task category selected was class payment. Therefore, the activity application program is the Class Payment application program. Thus, the Class Payment application program reveals the Class Payment activity window 2400. For this example, the next activity/task selected is the class ET201 for student DBS for the Class Payment activity. Referring to FIG. 28, after the user has completed the two next activities/tasks for the Class Payment activity, the summary To Do Category window 384 reveals that two Class Payment next activities/tasks exist and have both been completed (done).

Among other responsibilities, the administrator of the computer system of the present invention is responsible for the following. He or she defines entities (e.g., plants, sites, etc.); assigns USER(underscore)Ids to new users of the computer system of the present invention; assigns users to work groups; defines and maintains the security privileges of the users and work groups (e.g., which activities and next activities/tasks a user may access); disables and defines new next steps to comply with an organization's procedures; defines users and work groups responsible for performing next steps; changes the text of messages and other window text for each activity and next activity/task; and enters translated versions of messages and other window text for non-English speaking users.

The following illustrates how an administrator may define a new next step. As described above, for each activity, an event will be triggered. When an event is triggered, one or more next steps

will be selected which will result in a new next activity/task being added to a user's or work group's To Do List.

Referring to FIG. 29, the administrator accesses the Workflow Workbench activity from the administrator's list of activities (not shown). The Workflow Workbench activity window then displays an activity selected by the administrator, the activity application program name for that activity, all possible events for that activity, and the stored procedure name for each event in the WorkFlow Workbench window 2900. For this example, the Class Registration activity with its pam0500 activity application program name is displayed. For this activity, only one event exists. This event is "User registers for Class" (e.g. user adds a class) and the event identifier for the event is pamsam2ins.

Referring to FIG. 30, to add a new next step, the administrator may select, the "Zoom to Step and Assignments" option from a Zoom options list submenu (not shown). The user may access this option by selecting the Options mode and Zoom option from the Options mode list (not shown). The main application program then displays the Zoom mode list 3000 in which the "Zoom to Step and Assignments" may be selected. The Step and Assignments window 3100 is then displayed, as shown in FIG. 31, with the event selected.

Referring to FIG. 32, in the Step and Assignments window 3100, the administrator then selects an activity (next activity/task) application program from a list of next activities/tasks as the next step in the work flow process. For this example, the administrator has chosen pam0510, which represents the Class Payment application program. The administrator also selects the Enabled box to enable this new next step. Moreover, for this example, the Assignment Override box is selected by the administrator such that any user or workgroup information entered by the administrator will override any default values contained in the Class Payment application program. The main application program then sends the activity identifier for the next activity/task (ACTIVITY(underscore)ID) and event identifier (EVENT(underscore)ID) to a stored procedure. In one embodiment, the name of the stored procedure is psp(underscore)ins(underscore)nxtm. This stored procedure then creates a new row for the next step represented by its ACTIVITY(underscore)ID and EVENT(underscore)ID in the Next Step table 1225 (see FIG. 12). The stored procedure then sends a message to the main application program revealing that the row has been created.

Next, the administrator selects a message representative of a next activity/task category in which to identify the next activity/task in a user or work group's To Do List. Referring to FIG. 33, the administrator accomplishes this by selecting the Options mode and Zoom option from the Options mode list 2250. As shown in FIG. 34, the main application program then reveals a To Do Category window 3400 which displays the default next activity/task category message, which the administrator may modify if he or she chooses. The To Do Category window 3400 also allows the administrator to assign a system priority to the next activity/task category, and to select if the next activity/task category will be user defined. The Main Application Program then sends the activity identifier (ACTIVITY(underscore)ID) for the next activity/task, the event identifier (EVENT(underscore)ID), the category message identifier (MSG(underscore)ID), the message text, the user defined (USER(underscore)DEF) information, and the system priority information to a stored procedure. This stored procedure then stores the MSG(underscore)ID and

the USER(underscore)DEF for the ACTIVITY(underscore)ID and EVENT(underscore)ID in the NEXT(underscore)STEP table 1225 (see FIG. 12).

Finally, the administrator may assign the next activity/task to an entity, and a user and/or workgroup. Referring to FIG. 35, for this example, the administrator selects to assign the next activity/task to the default entity, represented as an asterisk (\*), and the sender (e.g., the creator of the next activity/task) who is in the Registration workgroup.

Referring to FIG. 36, as shown in the Workflow Workbench window 2900, the administrator may then add other next steps with corresponding assignments to a user or workgroup for the activity and each subsequent next activity/task. In this example, the administrator has assigned the Class Payment next activity/task to the Registration work group. Therefore, the work group identifier, represented as the MSG(underscore)GROUP(underscore)ID, is stored in the Next Step Options table 1230 (see FIG. 12). Other next steps selected by the administrator include the "Class payment type selected" (pamsam2up1) event with the next activity/task being "Registration Approval" (pam0520) which is assigned to user RSD.

The administrator may also activate the archiving feature of the computer system of the present invention. This feature may be used for backup and accountability purposes.

As shown in FIG. 37, a user may access a variety of additional features included with the computer/system of the present invention in the Preferences window 3700. These features include selecting how often the user would like to have his or her number of uncompleted next activities/tasks recalculated. This feature is called the "Refresh Task Counts" feature. Further, the next activity/task counts are recalculated at the specified time intervals selected by the user as described above for the Summary To Do Category window (see FIG. 7).

The features a user may select from also include activating a notification feature which notifies the user of new next activities/tasks which have been added to one of his or her To Do Lists and need to be acted upon. This feature is called the "Notify Me of New Tasks" feature. The user may then select to be notified with a message box in any window the user is currently viewing or with a certain number of beeps. The user may also select the next activity/task notification feature to be suppressed when the next activity/task is created by the user.

When the "Notify Me of New Tasks" feature is activated, the main application program sends the user's USER(underscore)ID to a stored procedure at the time intervals specified by the user. In one embodiment, the name of the stored procedure is psp(underscore)sel(underscore)mque(underscore)list. This stored procedure then obtains the number for the user's personalized To Do List (TODO(underscore)FOLDERN0) from the USER(underscore)MASTER table 1110 (see FIG. 11) with the USER(underscore)ID. Next, the stored procedure accesses the MESSAGE-QUEUE table 1140 (see FIG. 11) and obtains the relevant information on the next activities/tasks in the user's personalized To Do List which have a MSG(underscore)STATUS equal to "new." Finally, information regarding these "new" next activities/tasks is sent back to the main application program, which notifies the user of them with the message box or the beep(s).



Further, the user may activate the "Retrieve Next Task After Update" feature. This feature performs relatively the same sequence of events as described for the Next Task feature of the present invention. However, by activating this feature, the Next Task application programs and stored procedures are automatically accessed after saving information for a next activity/task.

Moreover, the user may activate the "Delete Completed Tasks After Update" feature, which automatically deletes the next activity/task from the user's To Do List as shown in the Summary and Detailed To Do Category windows (see FIGS. 16H and 16L) when completed. When this feature is not selected, the Summary and Detailed To Do Category windows display the next activity/task as done or complete. As shown in FIG. 38A, the next activities/tasks may also be deleted manually by the user in the Summary To Do Category window 384 and Detailed To Do Category window 385 (see FIG. 38B).

In another aspect of the computer system of the present invention, the user may obtain detailed information on a particular next activity/task. As shown in FIG. 39, the user may select a next activity/task category from the To Do Category window 384.

The main application program then sends the USER(underscore)ID formatted as the OWNER(underscore)ID, the OWNER(underscore)TYPE, and the FOLDERNO for the To Do List to a stored procedure. This stored procedure then calls the Next Task stored procedure, which accesses the MESSAGE(underscore)QUEUE table 1140 (see FIG. 11) with the OWNER(underscore)ID, OWNER(underscore)TYPE, MSG(underscore)ID, and FOLDERNO. This Next Task stored procedure then selects the highest prioritized next activity/task for the next activity/task category represented by its MSG(underscore)ID. Finally, the Next Task stored procedure sends the ACTIVITY(underscore)ID for the highest prioritized next activity/task from the next activity/task category back to a stored procedure. This stored procedure then accesses the MESSAGE-QUEUE table 1140 to obtain information on the user who initiated the next activity/task, also referred to as the original owner (ORIG(underscore)OWNER); the time the next activity/task was created (CREATE(underscore)TIME); the work group (MSG(underscore)GROUP(underscore)ID) responsible for the next activity/task; the user (FROM(underscore)USER(underscore)ID) who initiated the next activity/task; and the activity or next activity/task (FROM(underscore)ACT(underscore)ID) in which the next activity/task was initiated. This information is then sent back to the main application program which reveals it in the To Do Informational window 4000, as shown in FIG. 40.

According to a further aspect of the present invention, the computer system processes and prioritizes next activities/tasks for a user based on predefined conditions set by the user. These predefined conditions and actions resulting from the conditions in which the user may activate are referred to as agents.

According to another aspect of the present invention, the computer system includes a job scheduler feature which allows the user to create, schedule and submit jobs to run automatically. A job is typically an executable program, a DOS batch file or the like and an example of a job is a month-end departmental report.

According to yet a further aspect of the present invention, the computer system includes a mail feature. This feature allows the users to perform mail related activities including sending mail to and receiving mail from other users of the computer system of the present invention.

According to still another aspect of the present invention, the computer system includes a product request feature. This feature allows users of the computer system to communicate electronically with the administrator of the computer system so that the user may ask questions, receive system updates related to the computer system, and the like.

According to another aspect of the present invention, the computer system includes components which may be used to support a variety of business-related activities. These components include a component which allows intelligent, high-speed access to data, and another component which provides decision-making analysis and support. Moreover, these components are designed for use in a variety of business functions including manufacturing, distribution, finance, and human resources.

The implementation of the present invention described above with respect to FIGS. 1 through 40 assumes a single instance of database tables associated with the definition and implementation of the workflow, as visually depicted in FIGS. 11-15. The database tables shown in FIGS. 11-15 are commonly referred to as a "table family", because they exist and operate together as a single unit within a single database. The data in tables in such database which is used directly by the present invention to implement the new workflow techniques disclosed in this specification (e.g., in FIGS. 11-15) is generally known as the "platform table family" and the aggregate of this platform data in tables for a particular implementation of the present invention is therefore known as the "platform data". In a preferred embodiment, the present invention may utilize six table families, including distribution (ctlg), desktop (wijt), workflow (wact), message (mesm), product support (supt), and language (lang) data.

Similarly, data which corresponds to an application which resides in a database anywhere in the computer system (such as reference numeral 120 of FIG. 1A) is known as an "application table family", and the aggregate of such data for such an application is known as the "application data.

While a table family (defining a set of separate and distinct workflows) can be distributed among multiple servers, replicated among multiple servers, or reside on a single server, all tables in the family must have the same distribution properties as one another (distributed, replicated or centralized), according to the implementation of the present invention described previously with respect to FIGS. 1-40. All users of the particular workflow defined by the tables in FIGS. 11-15 preferably may make their primary database connection to the server which contains a "distribution catalog". The distribution catalog is the "address book" which all applications use to find the physical location of any table family.

Problems with this implementation, where a particular table family, and the workflow it defines, must reside on one server, include:

\* Single point of failure: The platform server becomes a single point of failure. If the platform server is down, then all users are down.

\* Performance/connection bottleneck: Since all users must connect to the server containing the distribution catalog, then this server becomes a connection bottleneck.

\* Network Performance: Since all users' desktop and workflow data must be on a single server, some users may be forced to work off of a server which could be geographically remote even though their business data may be on a local server. For example, a user in Houston may need to be directly connected to the Chicago server (location of the distribution catalog), while other data relevant to the running application(s) may reside back at the Houston server.

In order to overcome these and other limitations, in one embodiment, the present invention may be designed to define an approach for implementing the workflow techniques, previously described, on multiple servers to provide the following benefits:

\* Workflow across servers: That is, users and workgroups from different servers can participate in workflows.

\* Distribution-enabled applications: Applications may be designed to take advantage of the replication and distribution of data.

\* The ability to define the server and database location of the desktop for each user: The user's data (for example, desktop, To Do's and potentially application data) can be located close to the user to increase performance and availability.

\* The ability to define the server and database location of the workflow To Do tasks for each user and workgroup: For users, this may be the same location as their desktop.

\* Removal of the restriction that all users that participate in a common workflow must have their desktop on the same server: Desktops that participate in a common workflow may span servers.

\* A replicated copy of the distribution catalog on every server that contains desktop data: This will enhance overall system performance due to the fact that these tables are heavy "read-only" tables.

\* A replicated copy of other platform data on servers as required: This will increase data availability and will address the single point of failure issue. This benefit, along with the previous one, will provide site autonomy for each location in a computer network.

\* Client primary connection is to the database server containing their desktop: This should improve performance. Currently, in a multi-server environment, a client's primary connection could be to a server other than the server containing their application data. This can cause performance problems.

The primary means for accomplishing these objectives is through the use of data distribution and replication. The concepts of replicated and distributed tables will be discussed at a later point in this specification. To achieve this objective, the following capabilities may be included:

\* An enhanced procedure for each user to log onto the computer system in a way such that the computer system recognizes the users "desktop server".

\* Provide the ability for an application to determine where the primary copy of a table family exists.

\* Provide the ability to replicate tables to remote sites and propagate updates from the primary site to remote sites.

\* In a preferred embodiment, the solution should put minimal additional, visible constraints on a single site installation.

Prior to further describing the distributed design of the present invention, several terms may be defined. The definitions below are not meant to be an exhaustive interpretation of the related terms, but are merely provided as a basic definitional starting point for these terms. Those of ordinary skill in the art will readily recognize the scope of the meaning of these terms.

The following definitions are provided:

data replication versus distribution: There are at least four types of tables supported by the present invention. Tables can be either replicated, distributed, or a combination of both. Reference is made to the following table (Table A) for the appropriate terminology used when tables are distributed and/or replicated.

Thus, if a table is replicated, but not distributed, it is referred to as "centralized replicated". If a table is replicated and distributed, it is referred to as "distributed replicated". For purposes of the present invention, while distributed replicated tables can be implemented, they are not supported in a preferred embodiment. If a table is not replicated, and not distributed, it is referred to as "local", because in this case it is local to the user's computer. Finally, if a table is not replicated, but it is distributed, then it is referred to merely as "distributed".

replicated table: A replicated table is a table which has copies on multiple servers in the system. With the "primary copy replication algorithm" (PRCA) used according to the present invention, modifications are allowed only to the primary copy of the table and reads may be performed on the subscriber copies. Subscriber copies receive their data from the primary copy located at the primary server via the replication service.

"Fully replicated" means that copies of replicated tables are at every server site in the system.

"Partially replicated" means that there are copies of the replicated tables at some, but not all, server sites. For purposes of the present invention, a partially replicated scheme will be assumed.

distributed table: A distributed table is one where different pieces of the table reside on different servers. If a table is horizontally distributed, it is distributed by rows. What rows of a table go on what server may be determined by the "distribution entities" of the table family that the table belongs to. If a table is vertically distributed, subsets of columns reside on different servers. In one embodiment of the present invention, vertical distribution is not utilized.

An example of distributed tables in the present invention are the desktop tables in table family wjit. These tables include WJ(underscore)DRAWERS, WJ(underscore)FOLDERS, WJ(underscore)FOLDERDCMTS and WJ(underscore)DCMTS. Local updates to these tables do not need to be propagated to other servers.

Distribution of tables is accomplished through the use of "distribution entities". Distribution entities may be implemented as a new column on those tables which need to know where a table is. For example, a new column on the User Master table (reference numeral 1110 in FIG. 11, and Table D, described further below) and the Workgroup Master table (reference numeral 1210 in FIG. 12, and Table E, described further below) may be included, which points to the particular server where the subject table resides.

primary copy: This is the controlling, definitive copy of the table (or row of the table, etc.). All subscriber sites (see "subscriber copy" below) will receive updates based on the content of the data present in the table at the primary site. All data modifications (updates, deletes, or inserts) required for a table that is a replicated table will be performed against the primary copy. The server on which the primary table lives is the primary server.

subscriber copy: A replicated copy of the primary copy of a table (or row, etc.) which is read only. Subscriber copies receive updates from the primary copy via the replication service.

primary server: The first server on which the present invention is implemented. In one embodiment, the primary server contains the primary copies of all the replicated platform tables.

subscriber server: A server on which the present invention may be installed subsequent to its installation on the primary server. Replicated copies of platform tables reside on subscriber servers and updates to these replicated tables are propagated from the primary copies at the primary server site.

distribution-enabled: A property of an application that enables it to use local replicated data as implemented as part of the present invention (i.e., in an environment with distributed desktops and replicated platform tables) rather than relying exclusively on the primary copy.

distribution-compatible: A property of an application that enables it to operate properly with the present invention but not to be necessarily distribution-enabled.

distribution entities: The column that a table family is horizontally distributed by is the distribution entity. An enterprise's values for the distribution entity are the distribution entity values.

For example, a particular application, such as an application for managing manufacturing within an organization, may have its data distributed by a distribution entity called "site". Any offices or plants in different locations, for instance Chicago, Boston, and Atlanta, may be the distribution entity values for the distribution entity called "site".

In one embodiment, a table family can be distributed by 0, 1, or 2 distribution entities, although this is merely one implementation and not a limitation on the invention. The number of distribution entities is typically defined by the application; it is generally not user defined. Of course, the present invention may readily be implemented so as to allow the user to define the number of distribution entities.

**consolidated:** An attribute of a distributed table which means that the table needs to be queried across servers. An example of this type of table is the message queue. Generally, there will be a message queue per server that supplies workflow messaging for the local site.

If the organization implementing the present invention wanted to perform analysis on its workflow, a consolidated message queue that holds all messages across the organization would provide this type of information. This concept of a corporate-wide consolidated message queue is not supported in a preferred embodiment of the present invention, but, of course, may be readily implemented if desired, according to techniques known in the art.

**server initialization:** Server initialization is the process of installing the minimum platform services and objects on a server so that it can be used as a remote server.

**server materialization:** Materialization is the process of how subscriber servers are installed after the primary server is installed.

The current installation process for using the distribution and replication aspects of the present invention generally remain unchanged from that described previously for the basic workflow system of the present invention. When a subscription server is materialized, the source of the environmental data for this server may be from the primary server.

**server dematerialization:** Dematerialization is just the opposite of materialization as the term suggests. It is concerned with how a server is removed from service after it has been in use for a period of time. The primary concern here is what to do with the data and how does the user migrate the data to other servers.

**application platform data (APD):** Data added to the platform tables by applications. This is relevant with respect to installation and migration.

**table family:** A table family consists of a group of logically related tables that are handled as an atomic unit with regard to installation, replication, distribution, transaction management and referential integrity edits. Specific rules for table families according to one embodiment of the present invention include the following:

- All tables in a family must be installed on the same server and in the same database.
- All procedures and triggers associated with a table family can only update tables within that family. In other words, a procedure (transaction) cannot perform cross-family updates. In a further embodiment, and in order to support a distributed design, a table family generally cannot

contain tables of mixed types as defined above. Specifically, a table family cannot contain both replicated (read only) and local (local update) or distributed tables.

In some cases there may be dependencies between table families. For example, table families might need to be installed in the same database on all primary and subscriber servers. The install and materialization processes may be designed to enforce these dependencies.

The following are constraints or restrictions that are generally the result of the design of the distributed/replicated aspects of the present invention, in one embodiment of the present invention:

- \* No support is provided for distributed, replicated platform data, although this may be used for application data.

- \* All replicated platform data must be replicated to each storage facility (site). Data cannot be replicated at some subscription server sites and not at others. Again, appropriate design may eliminate this constraint.

- \* It is generally not possible to dematerialize (described further below) the server which contains the primary copies of the platform tables (without destroying the entire installation), or to change which server acts as the primary server. However, a tool could readily be provided to do this.

- \* All tables in a table family are generally distributed/replicated in the same fashion. This restriction reflects the decision to only record distribution/replication information at the table family granularity, but this restriction could be eliminated with an appropriate design change.

- \* Replication of text data is not supported. The replication mechanisms utilized in a preferred embodiment (Sybase Replication Server and a proprietary asynchronous RPC facility) will not work when text is involved. Of course, this constrain may also be eliminated with an appropriate replication mechanism that doesn't have this restriction.

- \* If a user is a member of a workgroup whose `wijt(underscore)location` is a different server than the user's desktop (the message queue for that workgroup is on another server) and that server is down, the user cannot participate in workflows that involve the workgroup.

- \* This design does not currently support the uniting of two distinct installations implemented according to the teachings of the present invention. It will only support materialization of new servers within one installation. Of course, appropriate design changes could be implemented to eliminate this constraint. Application table families can only be installed on primary or subscriber servers (i.e., only those on which that platform has been instantiated).

- \* Updates to replicated tables are typically (but not always) infrequent and generally involve a single row at a time.

- \* Update transactions that span multiple tables and include updates to replicated tables are typically rare and can be handled on a special-case basis.

\* The primary copy of a table is the default value for a server entry in tsdx (see Table C) that does not contain a from(underscore)server value.

\* The distributed design for the present invention does not, in one embodiment, include a general data movement tool for moving application data from one server to the other. Applications will have to develop data movement mechanisms on their own using the platform process to move users from one server to the another as a model. Of course, a general data movement tool may readily be designed.

\* The distributed design of the present invention does not, in one embodiment, include any features to facilitate distributed reporting. However, such features may also be readily implemented. Of course, the above constraints are provided merely to set forth one way in which the present invention may be implemented. These constraints are not limitations on the possible range of functionality of the present invention, but merely one design approach.

As described above, the present invention may be designed to support three types of servers: primary servers and subscriber servers. Each installation of the present invention generally may contain a single primary server where initial product installation and overall system administration is performed. This server is where the present invention is initially installed. Most administration tasks such as adding a new user, installing a new product, or changing a workflow definition require the availability of the primary server. FIG. 1E illustrates a configuration of the present invention in a relatively simple form -- where a primary server 110 and end user workstations 120 are utilized -- but no subscriber or remote servers. In this case, databases 160 (in this example, general ledger and accounts payable databases -- of course, these are just used as examples, and any type of database may be utilized) reside on the primary server 110, as does the operating environment 170 component of the present invention and ARPC 150. The elements of the system of FIG. 1E otherwise generally correspond to similarly identified elements depicted in FIG. 1A, described previously with respect thereto.

FIG. 1F depicts the other supported configuration, a configuration consisting of a primary server 110 and one or more subscriber servers 111. Subscriber servers 111 contain subscription copies of the platform configuration table families 131 and an instance of the user desktop table family 132 (described further below). Individual users or workgroups (and hence their respective end user workstations 120) can be assigned to a subscription server 111. A user 120 assigned to a subscriber server 111 can operate when the primary server 110 or the communication link 140 between the subscriber server 111 and the primary server 110 is down (inoperable).

The platform data 131 is divided into three categories, basic platform services 131A, the user desktop data 131B, and the platform configuration data 131C. The basic platform services consists of the arpc table family. This distributed table family must be installed on every server (110, 111, , etc.) in the system that has any application data relating to the present invention on it as part of the server initialization process (described further below).

The user desktop data 131B contains the user's drawers, folders, and attachments (described previously). This data (table families wjt and supt) may be horizontally distributed to every subscriber server 111 in the system based on the assignment of users 120 to servers. A particular



user 120 will be assigned to a server 111 when their user account is set up. A procedure may be provided to move a user from one server to another.

The platform configuration data 131C consists of the distribution catalog, activity definitions, security definitions, workflow definitions, message master, browse and file open parameters, and user and workgroup definitions (described previously with respect to FIGS. 11-15). This data is in table families *ctlg*, *wact*, *mesm*, and *lang*. The platform configuration data 131C may be centrally maintained on the primary server 110 and replicated to every subscriber server 111 in the system. The replication algorithm may allow "selects" from the copy on any server (110, 111, etc.) and "inserts", "deletes", and "updates" only to the copy on the primary server 110. Inserts, deletes, and updates may be propagated from the primary copy on the primary server 110 to the subscription copies on the subscription servers 111 automatically. Thus, the platform windows which maintain the data in the replicated table families (CTLG, WACT, LANG, and MESM) generally requires that the primary server 110 be available.

FIG. 1H illustrates the platform table families 131 divided into two groups: those that are replicated (CTLG 181, WACT 182, LANG 183, and MESM 184) and those that are distributed (WIJT 185, ARPC 186, and SUPT 187). Updates to the primary copy of each table (denoted with a trailing "p" -- e.g., 181p, 182p, 183p and 184p) are automatically propagated to the secondary copies (denoted with a trailing "s" -- e.g., 181s, 182s, 183s and 184s) for replicated tables, via replication services 140. Distribution of nonreplicated (distributed) data (e.g., 185, 186 and 187) may be handled administratively. Distribution and replication of platform data is independent of configuration and installation of applications.

To support replicated and distributed table families, a table distribution cross reference catalog TSDX (described below with respect to Table. C) may be created to allow applications to locate the primary (e.g., 181p, 182p, 183p, 184p, etc.) (for insert, update, or delete), the local subscription copy (e.g., 181s, 182s, 183s, 184s, etc.) of the table family (for select), or a local instance of distributed data (for insert, update or delete). Application programming interface calls (APIs) are available in the Catalog API that provide access to the TSDX table to support primary/subscription lookups.

In order to support workflows that span servers, the workflow engine described previously with respect to FIGS. 1-40 may be modified. A Workflow API may be devised to use 2-phase commit and/or ARPC when a workflow event needs to be moved from one server to another. The application architecture and the applications may be designed to use the workflow API wherever possible.

A distributed system of workflow requires several new administrative capabilities, in one embodiment. In order for the client to connect to all servers in a distributed system, the server and network address must be in the (SQLServer) section of the client machine's "win.ini" file (the "win.ini" file, or its equivalent, is used by the operating environment (e.g., Windows) to store information -- in this case information associated with the use of an SQL server-- about the operating environment). The logon sequence may be modified in order to make sure this server list is up-to-date. Specifically, in order to perform maintenance or install new products on an individual server, that server preferably should be made unavailable to individual users of the

present invention, while still being made available to the administrator of the system. A facility may readily be added to take a server or a database within a server off-line to users of the present invention.

The following table (Table B) lists the database tables that may be changed, and the new tables that may be created in order to support the distributed capabilities of the present invention. The left-most column of Table B identifies the name of the database table, the middle column specifies the family name of the table, and the right-most column of Table B describes the details of the change of creation.

The Distribution Catalog (TSDX) must be able to provide applications with the location of either the primary (maintainable) or subscription (read only) copy for a table family, depending on the intent of the application. If the application is performing a query, then it needs to know the location of the local subscription (or the closest subscription if there isn't a local one).

To support these requirements, the TSDX table structure may include two key columns. First, a flag column indicating whether an entry is a primary or subscription copy. Second, a "from(underscore)server" column, which identifies the location of the subscription copy of a specific table family for a client connected to the "from(underscore)server". The additions to the TSDX table are shown with sample data in each field in the following table (Table C): In this example

- \* A client connected to server FRM performs queries of table family ctlg on the FRM server (row 1).
- \* A client connected to server ATL performs queries of table family ctlg on the ATL server (row 2).
- \* A client connected to any other server performs queries of table family ctlg on the FRM server (as FRM is the primary server) (row 1). When looking up an entry in the TSDX table (Table C) with the intent of query, we first attempt to find a "from server" entry for the current server. If the entry is not found, then the entry with an \* is used.
- \* All clients perform updates of table family ctlg on the FRM server (row 1) because FRM is where the primary copy is located.

Where the ARPC database and all its objects are not installed on every server, an ARPC flag may be added to the TSDX table (Table C) to indicate that the table family uses the ARPC facility and therefore needs the ARPC objects installed on their server and database. This flag may be useful for knowing where the ARPC objects are for maintenance purposes. Of course, if the ARPC database and all its objects are installed on every server, then this flag is not necessary.

As discussed previously, a user's desktop information will be stored in one server/database location. Also, the user's workflow To Do messages must be transported to this location. The same is true for Workgroup To Do messages. A workgroup does not have a desktop, but its

messages must be transported to one location for all members to share. The present invention may be implemented to support one or more members of a workgroup being on different servers than the workgroup To Do messages, but in this case, the remote user will be impacted by performance and network availability problems on the workgroup server.

To support this situation, a "wjt server" column may be included in the User Master table (described briefly above with respect to Table B, and below with respect to Table D) and a Workgroup Master table (Table E, below) defining which instance of the "wjt" table family contains the desktop and workflow information. The value of the "wjt server" column must match a value defined in the distribution entity 1 column for the "wjt" table family in the TSDX table. This link to TSDX will allow services to find the physical location of the data.

In a preferred embodiment, the User Master table may be defined (in part) as indicated below in Table D: The User Master table of Table D corresponds to the User Master table shown as reference numeral 1110 in FIGS. 11 and 15, and illustrates two additional columns which may be added to that table.

As illustrated in Table D, the User Master table may include a column identifying a User ID, and a column identifying which server contains the "wjt" table family which includes an instance of the user's desk and workflow information.

Likewise, a Workgroup Master Table may be defined (in part) as indicated below in Table E: This table is similar to the User Master table of Table D (and reference numeral 1110), except that the Workgroup ID for a group of users is identified in a first column, rather than a specific user ID.

Some columns in the User Master table (reference numeral 1110 in FIGS. 11 and 15, and further defined in Table D) in the current implementation need to be updated whenever the user logs onto the application of the present invention. Additionally some new user configuration data is required to support the new sequence number generation algorithm of the present invention (described further below). Since the user needs to be able to logon when the primary server is down, these columns need to be on a table which is distributed to the desktop server (110, 111 or 112, as appropriate). Thus a User Configuration Table, shown below as Table F, may be added to the wjt table family to hold these values. This table may contain the following columns:

Likewise, a Workgroup Configuration table may be created on a distributed desktop server, similar to the User Configuration Table, except corresponding to workgroups, as opposed to individual users. The Workgroup Configuration table may contain the following columns, as illustrated below in Table G:

A Network Access table may also be created on a distributed desktop server 111, which defines the connection strings that clients need to have in their "win.ini" file to connect to every server in the system. There is a row in this table for each server/network type pair at the site. This table may contain the following columns, as illustrated below with respect to Table H:

For most sites which have standardized on a single client TCP/IP stack, this table will contain a single row for each server. Part of the client logon sequence may be to query this table to make sure its "win.ini" file has all the up-to-date server strings. Accordingly, the particular string added to "win.in" may have the following format:

<Server>=<Network Type>,<Connection String>

where the name of the server is substituted for <Server>, the network type is substituted for <Network Type>, and the appropriate connection string is substituted for <Connection String>.

The present invention may be designed to allow network connection strings to be stored in a shared file on a file server rather than in the "win.ini" file. Thus, if the present invention is implemented to use such a scheme, the need for the Network Access Table can be eliminated.

When a new server is instantiated, accounts generally need to be created for every user in the system on that server. The following table (Table I) may be created to contain a list of user/server/password combinations on the new server until the user logs on and the password is changed. Table I may contain the following fields:

I

The initial installation process to install the present invention on a single server can remain essentially unchanged from the general configuration described previously with respect to FIGS. 1-40. However, new installation scripts may be developed for server initialization, distributed desktop option installation, and subscriber materialization.

In a multi-server configuration, the application install process will first install the Application Platform Data 131 on the primary server 110. The replication system 150 will automatically propagate these additions to any subscription servers 111. The application database 131 can then be installed on any server in the system that has been initialized.

The install process required for the distributed aspect of the present invention also generally requires specific installation scripts. The installation of maintenance stored procedures (insert, delete, and update), loading of initial data, and platform migration may be part of the primary server installation, while the installation of table definitions and select stored procedures may be part of the primary installation and subscriber materialization.

As depicted in FIG. 41, an installation script may be created for server initialization. With reference to FIG. 41, this script will do the following:

- 1) (step 4101) Load system stored procedures necessary for the present invention into the master database (e.g., on server 110, 111, etc.).
- 2) (step 4102) Create the logical server names required for the applications.
- 3) (step 4103) Create an instance of the ARPC database 150 on the server.

- 4) (step 4104) Register the instance of the ARPC database in TSDX (see Table C).
- 5) (step 4105) Add the entries into the Network Access table (see Table H) for the server so that clients will be able to get the network connection information for the server.
- 6) (step 4106) Create all the necessary Sybase user accounts on the server for existing users (make sure all passwords are synchronized system-wide, in a preferred embodiment). Application databases will only be able to be installed on servers that have been initialized.

As illustrated in FIG. 42, an installation script may be created to install the distributed desktop option on the primary server 110. This will install the components on the primary server 110 that are required to support subscriber servers 111. With reference to FIG. 42, this script will do the following:

- 1) (step 4201) Install Sybase Replication Server.
- 2) (step 4202) Create a Replication Server 150 and a Log Transfer Manager 151 on the primary server 110.
- 3) (step 4203) Create the DBSReplicate user on the primary server 110, the Log Transfer Manager 151, and the primary server Replication Server 150.
- 4) (step 4204) Create the replication definitions for tables that are replicated.

As illustrated in FIG. 43, an installation script may be implemented to create a subscriber server 111. A prerequisite to running this script will be running the initialization script for the server as described above and installing the distributed desktop option on the primary server 110. With reference to FIG. 43, this script will do the following:

- 1) (step 4301) Build the databases and their objects on the subscription machine 111 with no initial data and leaving out the stored procedures to update, delete, or insert data into replicated tables.
- 2) (step 4302) Create the DBSReplicate user 153 on the subscription SQL Server 111.
- 3) (step 4303) Set up security so that the DBSReplicate user 153 can insert, update, and delete data from the replicated table families.
- 4) (step 4304) Create the subscriptions with the proper options so that Replication Server 150 does the initial data copy.
- 5) (step 4305) Update TSDX (Table C) to have the entries for the new subscription copies of the replicated table families and the new instance of the wijt table family.
- 6) (step 4306) Create user accounts for all existing users on the server. Since materialization process does not know the correct password, it may generate a randomly selected password,

encrypt it, and store it in the Server Temp Passwords table. The user logon process will change the password and delete this record the next time the user logs onto the present invention. After a server is materialized existing users can be moved to the server, or new users can be assigned to the server.

Dematerialization is the process of removing a server (110, 111, 112, etc.) from an installation of the present invention. Note that a server may contain no users or workgroups and still function properly to support applications. Such a server is not considered dematerialized.

FIG. 44 depicts the steps which may be performed as part of the dematerialization process. With reference to FIG. 44, these steps are described below:

1. (step 4401) Remove all users and workgroups from the server, either by deleting them or moving them with the administration tool used to move them to the server originally.
2. (step 4402) Remove all application data and applications from the server. Providing tools and instructions on how to move application data to another server is generally the responsibility of the applications.
3. Begin dematerialization:
  - a. (step 4403a) Update TSDX (see Table C) on the primary copy server 110 to indicate the dematerialized server is no longer available.
  - b. (step 4403b) Check to see that the ARPC queue (e.g., 153) on the dematerialized server is empty. If not, disallow dematerialization (step 4404).
  - c. (step 4403c) Check to see that the ARPC queue (e.g., 150) on the primary copy site 110 contains no updates routed to the dematerialized server. If not, disallow dematerialization (step 4404).
  - d. (step 4403d) Remove all databases on the dematerialized server.

In the event that the present invention is first implemented without those changes in the tables defined previously with respect to Tables A through I, then this configuration may be migrated to include distributed functionality by performing the steps illustrated in FIG. 45. With reference to FIG. 45, these steps are described below:

- 1) (step 4501) Add a User Configuration record for each user, as described previously with respect to Table B. This record will contain the MAPI logon information, ToDo Folder number, and enabled flag from User Master. The sequence number fields will be set to 0. Additionally the network type for each user will have to be initialized.
- 2) (step 4502) Add a Workgroup Configuration record for each workgroup, as also described previously with respect to Table B.. This record will contain the ToDo Folder number for the workgroup.

3) (step 4503) In a preferred embodiment, such a migration process does not migrate subscriber servers 111. Each subscriber server replication definition may instead be dropped and recreated as part of the process.

The install process described above may use a Sybase Replication Server to initially copy the replicated table families from the primary server 110 to the subscription servers 111. It will also take care of propagating any changes that are made to the tables when new products are installed or when the primary copy is updated by a maintenance window. Occasionally the system administrator may believe that the primary and subscription copies of a table family are not in sync. Symptoms of out of sync tables include:

- \* Workflow events are disappearing.
- \* Application Architecture windows are taking errors indicating that initial data is not present or is not correct.
- \* Activities available to users assigned on one server are not available to users assigned to other servers.

Scripts may be provided to verify and fix subscription copies of the replicated tables. For example, these scripts may call a "rs(underscore)subcmp" utility for each table family to verify that the subscription copies of tables are in sync, and may fix any tables that are not in sync.

In one embodiment of the present invention, the ctlg, wact, mesm, and lang table families (described previously) are replicated. In order to support replicated table families, the Distribution Catalog API (application programming interface) may be created to support locating either the primary or the closest subscription copy of replicated table families. Thus, calls to this API will locate the subscription copy of the desired table families. If so desired, This API may be designed so that additional parameters or globals may be passed to locate the primary copy of the desired table families.

In a preferred embodiment, where it is not possible to extend the API, utilized for the embodiment of the present invention defined with respect to FIGS. 1-40, to support replicated table families, a new API call may be defined that includes the primary/subscription flag. In this case, the name of the new API may be different than the name of the API for the basic invention. Thus, the basic API may be used to locate the subscription copy of the desired table family, but preferably in future generations of the present invention, the new API can be used for this purpose.

Again, the present invention utilizes the concept of replicated table families. An installation has one primary copy of a replicated table family which can be read or modified and multiple subscription copies of the table family which are read-only. When an application needs to modify data in a replicated table family it must find the location of the primary copy of the table family. When an application needs to read data from a replicated table family it needs to find the location of the closest subscription copy of the table family. The existing table family catalog

APIs may be modified according to the teachings of the present invention to support locating either the primary or the closest subscription copy of a replicated table family.

The table below (Table J) lists the new and modified Powerbuilder (the development tool available from Powersoft Corporation) functions and Stored Procedures (such as SQL stored procedures) forming the Distribution Catalog API that look up table and table family location information in the TSDX table. Listed for each is the name of the API, the name of the API it replaces, the type of the API, and a description of the API.

Where possible the existing API is extended to support replicated table families. In this case the name of the new API is the same as the name of the current API. Existing calls to the current API will locate the subscription copy of the desired table families. Additional parameters or globals can optionally be passed to these functions to locate the primary copy of the desired table families.

Where it is not possible to extend the existing API to support replicated table families, a new API call was defined that includes the primary/subscription flag. In this case, the name of the new API is different than the name of the current API to allow both APIs to be supported while the applications make the transition to the new API. where "PB" in the type column indicates that the procedure may be implemented by using PowerBuilder code, and "SP" indicates that the procedure may be implemented as a stored procedure.

Each Distribution Catalog API described above with respect to Table J may include an "Existing Function" section describing existing APIs which are related to the new API, as well as an "Application Impact" section describing the impact that the new API has on the implementation of the present invention as defined previously with respect to FIGS. 1-40.

In order to implement the replicated aspects of the present invention, based upon the teachings of the present invention defined previously with respect to FIGS. 1-40, all platform activities and browser code should be examined to find places where the ctlg, wact, mesm, and lang table families are used. If access is read-only, the code needs to be verified to make sure it is using the subscription copy of the table family. If access is required for insert, update, or delete, then the code to locate the table family will need to be implemented to locate the primary copy of those table families, and perform the particular operations thereon.

With respect to the user logon process, instead of each user 120 connecting to a single server 110, users will connect to their desktop server, which may be one of a variety of servers as depicted in FIGS. 1A-1H. This may be accomplished efficiently by storing the connection information for each user 120 in the user's DBS.INI file (191 in FIG. 1H), instead of the DBSERVER.INI file (192 in FIG. 1H). This information may be validated at login time. The DBSERVER.INI file, which is shared by multiple users on a file server, may still be used for initial logins, fallback logins and other shared information. The platform logon procedure may readily incorporate these changes to make them transparent to applications built with the application architecture.



When implementing the invention described with respect to FIGS. 1-40, users 120 connect and login to the TSDX server and database defined in the DBSERVER.INI file, which is shared by many users on a file server. The contents of a typical DBSERVER.INI file is shown below: Since each user will instead be connected to their "wijt server", the login procedure will change as follows.

For a user's first login, the initial connection to the server and database defined in the DBSERVER.INI file may be used. This server and database will be the location of the primary Distribution Catalog. However, after the connection is made, the login process may look up the user's "wijt(underscore)location" (desktop server) in the User Master table (see Table D) (this table is replicated to all subscription servers). As shown in the following example, three new items may be stored in the user's DBS.INI file, which are:

For a user's subsequent logins, the initial connection would be made to the WijtServer and CtlgDatabase defined in the DBS.INI. After logging in successfully, these "wijt" values would be validated against the User Master (see Table D) and TSDX tables (see Table C) in the catalog database of the wijt(underscore)location server in order to check if the:

- \* User's desktop has been moved to another server or database
- \* DBS.INI file has been "corrupted".

Additionally, the logon process may check to see if any records for this user exist in the Server Temp Passwords table (see Table B). If so, the process will logon to the indicated server, change the users password to the correct password, and use ARPC to delete the Server Temp Passwords table entry. This process is the first thing executed after sign-on so that if the user has been moved to the new server, the password will be correct by time the logon process tries to directly log onto the server.

All of the above processing may be accomplished via the processing illustrated in FIG. 46, and described in the pseudo-code below:

Once the user client 120 has connected to the correct server, it may do a select on the Network Access table (see Table H) to get a list of the servers and network address information for its network type. It may also make sure that each server has an entry in the local win.ini file.

In order to support movement of users 120 from one server to another, drawer, document and template numbers may be unique across all servers. To implement this requirement in the present invention, the following two tables (Tables K and L) may be created: In the preferred embodiment, the calculation of a new drawer, document or template number for the user 120 for reference purposes may be accomplished using a combination of server number + sequence number (from Tables K and L) within the server. Sequence numbers within a server are allocated sequentially and are never reused. Server numbers are allocated sequentially and never reused as well.

In order to prevent the table which contains the sequence number for the servers (referred to as the WIJ(underscore)SEQNO table, described previously with respect to Table B) from becoming a bottleneck, when a local user allocates a sequence number they are given a range of 10 numbers which are stored in the User Configuration table (Table F). Thus, until they use up those 10 sequence numbers they only access the User Configuration table to get a new number.

In addition to utilizing the sequentially allocated sequence numbers for the servers, the present invention may also utilize a there will also be the addition of the WIJ(underscore)NEXT(underscore)SERVERNO table (also described previously with respect to Table B) on the primary server 110. This table will contain the server number of the next server number to be materialized.

FIGS. 48 describes a series of process that may be used, in one embodiment:

At installation:

- \* (step 4801) Set up WIJ(underscore)NEXT(underscore)SERVERNO on primary to contain 30,000,000.

- \* (step 4802) Set up WIJ(underscore)SERVERNO on the primary to contain 20,000,000. At server materialization:

- \* (step 4803) Set up WIJ(underscore)SERVERNO.SERVERNO to contain the server unique sequence number from WIJ(underscore)NEXT(underscore)SERVERNO. SERVERNO.

- \* (step 4804) Add 10,000,000 to WIJ(underscore)NEXT(underscore)SERVERNO.next(underscore)serverno.

- \* (step 4805) Set up WIJ(underscore)SEQNO.seqno on the new server to contain a single row that is 0. When a user is created:

- \* (step 4806) Set user(underscore)config.next(underscore)seq(underscore)no and user(underscore)config.last(underscore)seq(underscore)no fields to 0

FIG. 49 depicts a process for generating a unique number for use with respect to the process of FIG. 48. With reference to FIG. 49, this process is described as pseudo-code below, where standard conditional, variable assignment and parameter returning logic is shown:

If the sequence number table (WIJ(underscore)SEQNO) ever overflows producing an error, then a new server number can be allocated for the server. If the number of servers (212 servers) supported by this algorithm is ever threatened to be overflowed, then the sequence number column of the WIJ(underscore)SEQNO table can be changed from an integer to a decimal column. This change would, of course, require coding and table changes, but the migration to this approach would be trivial.

Implementing the process described above is straight-forward. The new WIJ(underscore)SEQNO and WIJ(underscore)NEXT(underscore)SERVERNO tables need to be added to the install process. The SEQNO#getnext(underscore)I stored procedure may readily be changed to implement this new algorithm. The install process needs to set up WIJ(underscore)SERVERNO on the primary server to 20,000,000. Finally, the correct user(underscore)config records need to be built as part of migration for existing users.

The User Configuration table (see Table F) may contain the enabled flag, the mail logon information, the ToDo Folder number, the network type, and the sequence numbers allocated for a user 120. The platform login sequence may readily be modified to use the enabled flag and the mail logon information from this table rather than from the user master table. The workflow engine (described previously with respect to FIGS. 1-40) may readily be modified to use the ToDo folder number from the User Configuration Table rather than from the user master table. The sequence number generation algorithm may use the sequence numbers stored in the user configuration table to allocate sequence numbers as described previously.

The product request feature of the present invention may also be distributed. The Product Request windows may simply add the server name as parameter. In one embodiment, users 120 may only be able to view Product Request information for a particular server with a single query.

With respect to the implementation of workflow in a distributed environment, the separation of the WACT and WIJT tables into separate families and the distribution of some table families make it necessary to identify all situations in which modifications are being made to platform data to guarantee the updates are safe and correct. Some situations may require the introduction of ARPC or two-phase commits by the client. For example, the Trigger Workflow event may be modified to use the ARPC utility to send To Do messages to the user or workgroup's "wijt(underscore)location", and Reassign To Do will require a two-phased commit when the destination owner is on a different server than the source owner.

To reduce the number of places where updates need to be made to the workflow techniques described with respect to FIGS. 1-40, to support workflow distribution, wherever possible it is preferable to make use of a new Workflow API. The Workflow API will shield platform and application code from making platform table updates directly. The assumption underlying the design of the distributed workflow system of the present invention is that applications either do not make updates to tables, or they make updates in limited situations and can frequently rely on the Workflow API.

The following paragraphs discuss the workflow API functions that may potentially "touch" instances of the wijt table family on multiple servers. These functions will require two-phase commit, Asynchronous RPC, or both:

WFDelTasksByFromAct() and WFDelTasksByToAct() functions. These functions need to delete tasks from the consolidated message queue, i.e. the message queues on every machine in the system. In a distributed system this may typically only be done synchronously by two-phase commit on every server in the system. Since a synchronous delete would require the availability of all servers in the system, these functions may be supported using ARPC.

Certain applications (such as "Financial Stream", available from Dun & Bradstreet Software Services, Inc., the assignee of the present patent application) uses these functions at the start of some of their batch programs. These programs generate a a lot of events. When the program is run, any events that were generated by a previous run of the program are no longer needed. Thus, when the batch program starts they want to delete the events that were generated by the previous run of the program. All such events have the same from(underscore)activity, so they may use the `psp(underscore)del(underscore)mque(underscore)by(underscore)from(underscore)act` to do this delete.

In one embodiment of the present invention, message queue entries may have the key values from the calling activity separated from the data which is passed onto the next activity in the workflow. The functions which do asynchronous deletes will take these key values as input. Thus the batch programs noted above will have to generate a key (time of day perhaps) each time they run and save it in the database. The next time the program runs it can do the deletes based on the key value. As long as the key is properly generated and saved, this implementation will prevent race conditions between ARPCs and newly generated ToDo messages.

`WFReassignTask()` and `WFReassignAllTasks()`. These functions should attempt to reassign the tasks using 2-phase commit. If the Workflow API cannot login to the target server, the tasks should be reassigned via ARPC.

These functions will also have to be modified to handle duplicate keys. Since the unique timestamp is part of the key it is possible that an identical workflow event exists on the destination of a reassign. The stored procedures that handle re-assign on the destination will have to handle duplicate key errors and generate a new key when they occur.

`WFTriggerEvent()`. The `WFTriggerEvent()` function generates a workflow event. This event may be targeted to a user or workgroup on the local server or on the remote server. The stored procedure that supports this API function may be changed to generate an ARPC if the targeted user or workgroup is on the remote server.

`WFRegister(underscore)ComDB()`, `WFRegister(underscore)DBLIB()`, `WFRegister(underscore)New()`, `WFRegister(underscore)PB()`. In order to do any operations with 2-phase commit the workflow API will have to have the ability to logon to a new server. To log onto the new server, it will require the user's password. Thus the Workflow Register functions must be modified to take the user's password as an input parameter.

In order to administer the present invention, a User window may be included in the distributed implementation to include the server that the user is assigned to and the network type that the user uses. The server field may be entered when a new user is created. For existing users this field may be display only on this window. The separate Move User activity can be used to move a user from one server to another. The User Window will also have to be modified to add and update both the User Master and the User Configuration records for a user when a user is added or changed. This operation will require two-phase commit because the two records may be on different servers.

A Workgroup Window will require the same changes as the User Window. The server the workgroup is assigned to may be added to the window. Two-phase commit may be required to update the Workgroup Master and the Workgroup Configuration records when a workgroup is added or changed. A Move Workgroup window may be added to move a workgroup from one server to another.

A new User Movement activity may be required to move a user from one server to another. This process includes moving the user's data to the new server, copying the User Configuration record from the old server to the new server, and changing the User Master record to point to the new server. The details of this process are described below with respect to FIG. 50:

- 1) (step 5001) Get the user 120 to log off.
- 2) (step 5002) Disable the user so they can't re-log on.
- 3) (step 5003) Update user(underscore)master (Table D) to point to the new server and update user(underscore)config (Table F) on both the new server and the old server. These updates should preferably be made in one transaction.
- 4) (step 5004) Read through each record in each table (except the message queue) on the source server and copy it to the destination server. In one embodiment, this process may be a specialized C routine that logs onto each server to move the data.
- 5) (step 5005) Move each message in the message queue from the source to the destination server. For this process to work properly it preferably needs to use two-phase commit to move each message.
- 6) (step 5006) Re-enable the user so they can log on to the new server.
- 7) (step 5007) Have the user log onto the new server and verify that everything was moved properly.
- 8) (step 5008) Delete the user's data from the old server. If anything goes wrong during this process of FIG. 50, the following steps may be taken to recover from the error. Reference is made to FIG. 51 when describing these steps:
  - 1) (step 5101) Delete all the user's data from the new server except for message queue messages.
  - 2) (step 5102) Start over from step 4 in the above sequence (FIG. 50).

As with the move user process of FIG. 50, a move workgroup process may be implemented according to the following steps, described with respect to FIG. 52:

- 1) (step 5201) Update workgroup(underscore)master (Table E) to point to the new server and update the Workgroup Configuration table (Table G) on both the new server and the old server. These updates preferably should be made in one transaction.

2) (step 5202) Move each message in the message queue from the source to the destination server. For this process to work properly it needs to use two-phase commit to move each message.

In addition to the APIs which operate on the TSDX table (See Table C), the Table Distribution Maintenance activity may be modified to support maintenance of subscription copies of replicated tables.

To take a server or database off-line from users of the present invention, an activity may be created to remove execute permission on all stored procedures and to remove select permission on all tables in the database or server.

To accomplish this, for each table family in the system there may be two new stored procedures. The Disable Permissions stored procedure may operate to revoke execute permission on all stored procedures in the table family from public and revoke the select permission on all tables and views in the table family from public. Conversely, the Enable Permissions stored procedure may operate to grant execute permission on all stored procedures in the table family to public and grant select permission on all tables and views to public. These stored procedures may be named so that the stored procedure name can be derived from the table family id. In a preferred embodiment, only a system administrator with sufficient privileges will be given execute permission to these stored procedures.

A new activity may be added to disable a database or a server. This activity may operate to build a list of stored procedures that it needs to run from TSDX. It may then run the stored procedures to revoke the permissions on all table families in the database or server. Again, only a system administrator with sufficient privileges should be able to use this activity.

An activity may also be readily created to enable a database or a server. It may call the stored procedures necessary to enable permissions on the stored procedures and tables in the affected families.

In a further embodiment, all the database access routines may be changed to check for a "no permission" error. If they get this error, then they should return a message indicating that the database is not currently available. Additionally the desktop code that accesses the "wjt" table family may be changed to gracefully handle the error and log the user off of the system present invention when the user's "wjt" table family is disabled.

The install process should preferably be designed to install all the table families in a disabled state. Thus, any statements in table and stored procedure definition files that grant permissions need to be removed. This keeps users from trying to access a database while an installation is in progress. The last step in the install may optionally run the stored procedures to enable permissions on the tables and stored procedures.

ARPC may also have to be changed to handle a "no permission" error as a retryable error.

A new activity may need to be created to maintain the Network Access Table (Table H). This activity allows the administrator to view the current entries, create new entries, modify existing entries, or delete obsolete entries.

In order for applications to be compatible with the distributed aspect of the present invention, the guidelines listed below generally must be followed:

1) The wijt table family may be distributed in the present invention, based on the server that the user is assigned to. The following stored procedures in the wijt table family may potentially modify data on multiple servers, in one embodiment:

psp(underscore)del(underscore)mque(underscore)by(underscore)from(underscore)act

psp(underscore)del(underscore)mque(underscore)by(underscore)to(underscore)act(underscore)l

psp(underscore)upd(underscore)mque(underscore)reassign(underscore)l

psp(underscore)upd(underscore)mque(underscore)wrkgp(underscore)reassign

psp(underscore)trigger(underscore)ams(underscore)event(underscore)l

Any programs or windows in an application that call these stored procedures directly preferably should be changed to call the corresponding Workflow API. In a preferred embodiment, the WFDelTasksByFromAct() and FDelTasksByToAct() functions (which replace the sp(underscore)del(underscore)mque(underscore)by(underscore)from(underscore)act and psp(underscore)del(underscore)mque(underscore)by(underscore)to(underscore)act(underscore)l stored procedures used in the embodiment of the present invention described with respect to FIGS. 1-40) may be used.

2) As noted above the wijt table family may be distributed in the distributed implementation of the present invention. In this case, a

AM(underscore)Svr(underscore)Db(underscore)Owner(underscore)g Application Architecture global may contain the location of the user's instance of the wijt table family. Any program or window that uses stored procedures in the wijt table family that does not use the AM(underscore)Svr(underscore)Db(underscore)Owner(underscore)g global to find the location of the correct instance of wijt should preferably be changed to use the corresponding Workflow API function. This includes programs or windows that call the following stored procedures:

psp(underscore)del(underscore)mque

psp(underscore)del(underscore)mque(underscore)agt

psp(underscore)del(underscore)mque(underscore)by(underscore)own(underscore)fr(underscore)a  
ct

psp(underscore)del(underscore)mque(underscore)by(underscore)own(underscore)fr(underscore)c  
ol(underscore)l

psp(underscore)del(underscore)mque(underscore)by(underscore)own(underscore)gr(underscore)  
col(underscore)l

psp(underscore)upd(underscore)mque(underscore)msg(underscore)anystatus

psp(underscore)upd(underscore)mque(underscore)msg(underscore)status

psp(underscore)upd(underscore)mque(underscore)msg(underscore)status3

psp(underscore)sel(underscore)mque(underscore)date(underscore)msg

psp(underscore)sel(underscore)mque(underscore)msginfo

psp(underscore)sel(underscore)mque(underscore)next(underscore)msg(underscore)l

psp(underscore)sel(underscore)mque(underscore)next(underscore)step(underscore)l

psp(underscore)sel(underscore)mque(underscore)prior(underscore)msg(underscore)l

psp(underscore)sel(underscore)mque(underscore)step(underscore)msg(underscore)l

psp(underscore)sel(underscore)mque(underscore)step(underscore)msg(underscore)key

3) The wact, lang, mesm, and ctlg table families may be replicated according to the teachings of the present invention. The default behavior of the existing Catalog APIs may be to locate the closest subscription copy of these table families. Thus, programs or windows that call stored procedures that modify data in the wact, lang, mesm, or ctlg table families may have to be changed to use the new versions of the Catalog APIs to find the primary copy of these table families. This includes callers of the following stored procedures:

psp(underscore)del(underscore)ibpd(underscore)udak

psp(underscore)del(underscore)obpd(underscore)udak

psp(underscore)ins(underscore)ibpd(underscore)udak

psp(underscore)ins(underscore)obpd(underscore)udak

psp(underscore)upd(underscore)colm(underscore)udak

psp(underscore)upd(underscore)colv(underscore)udak

psp(underscore)upd(underscore)ibph(underscore)udak



psp(underscore)upd(underscore)ibpl(underscore)udak

4) The application installation process may be implemented according to the teachings of the present invention to support installation of basic platform services on all the servers in the system, replication of the replicated table families, and distribution of the workflow events and desktop data.

5) Each application table family may have to provide stored procedures to enable and disable permissions on tables, views, and stored procedures in the table family. The statements to grant permissions to objects may have to be removed from the file that creates the object.

In general, the applications must make the changes listed below in order to become distribution-enabled. An application that is distribution-enabled operates properly with the distributed implementation of the present invention and uses local copies of replicated table families rather than relying exclusively on the primary copy. The general change requirements for applications are listed below:

1) Application programs that directly read dbserver.ini (a local initialization file) to find the location of the catalog (TSDX) server (for example Query & Reporter) will have to be changed to use the platform logon algorithm to find the local server to log on to. Sample application programs in this class from the assignee of the present invention (Dun & Bradstreet Software Services, Inc.) include: the User-Defined Accounting Key (UDAK) customization application, Query & Reporter, and Management Reporter.

2) Application programs that obtain the name of the TSDX server from somewhere other than the dbserver.ini file will have to change to make sure that they are locating the subscription copy of the ctlg database on the server "closest" to their application data. Application programs, from the assignee of the present invention, in this class include the Job Scheduler, the Scheduler API, and InterQ.

3) Application programs that call the  
psp(underscore)del(underscore)mque(underscore)by(underscore)from(underscore)act or  
psp(underscore)del(underscore)mque(underscore)by(underscore)to(underscore)act(underscore)l  
stored procedures (described previously with respect to FIGS. 1-40) need to be redesigned to not use these functions. Applicable workarounds may be designed in this case.

4) Application programs or windows that call Distribution Catalog APIs that change in the distributed implementation of the present invention need to be changed to call the new APIs.

Finally, in various other embodiments of the present inventions, certain simplifications may be made. For example, if the Sybase CT-Lib library for enabling clients to communicate with servers is utilized instead of DBLIB in the present invention, the need for the Network Access Table and the associated maintenance GUI can be eliminated.

Conditional Work Flow (Background)

In a further embodiment of the present invention, various enhancements may be made in order to to improve upon the functionality of the work flow environment. Included among these enhancements is the ability to execute the work flow of the present invention based upon certain conditions. These enhancements, including conditional workflow, are briefly described below, and are described in further detail later in the specification. These enhancements essentially add functionality to the SmartStream Version 3.0 product available from Dun & Bradstreet Software Services, Inc., Atlanta, Georgia, the assignee of the present patent application.

The workflow engine described previously is implemented in the `psp(underscore)trigger(underscore)ams(underscore)event(underscore)1` stored procedure (see the appendices of co-pending U.S. Patent Application No. 08/213,022 and U.S. Patent Application Serial No. 08/475,575). This procedure takes parameters passed by the call, the column ids from `event(underscore)master`, the next activity from the `next(underscore)step` table, and user assignment parameters from the `next(underscore)step(underscore)options` table to generate a `ToDo` that is put on the message queue.

The `psp(underscore)trigger(underscore)ams(underscore)event(underscore)1` stored procedure takes the following parameters:

- \* `@p(underscore)event(underscore)id` - The event id of the event being generated. Every event in the system has a unique event id. In general an event is defined for every insert, delete, or update to a window in the system, although additional events can be defined.
- \* `@p(underscore)next(underscore)step(underscore)ent(underscore)val` - Single 30 character parameter used for conditional generation of the owner of the `ToDo` that is generated.
- \* `@p(underscore)from(underscore)user(underscore)arg` - The user id of the user generating the event.
- \* `@p(underscore)from(underscore)act(underscore)arg` - The activity generating the event. This parameter is just passed untouched to the message queue. It is not used by the algorithm that generates the next event.
- \* `@p(underscore)assign(underscore)to(underscore)arg` - This value tells how to assign the event if the `next(underscore)step(underscore)options.assign(underscore)to` is "D". It can have the following values:
  - \* "D" - ignore `to(underscore)owner` and take next owner from `next(underscore)step(underscore)options`
  - \* "U" - `to(underscore)owner` is a user
  - \* "G" - `to(underscore)owner` is a group

\* @p(underscore)to(underscore)owner - The user/workgroup to assign to if @p(underscore)assign(underscore)to(underscore)arg is not "D" and if next(underscore)step(underscore)options.assign(underscore)to is 'D'.

\* @p(underscore)msg(underscore)priority - The message priority. This parameter is just passed untouched to the message queue. It is not used by the algorithm that generates the next event.

\* @p(underscore)col(underscore)val(underscore)1 to @p(underscore)col(underscore)val(underscore)16 - Column values used to initialize the next event. These parameters are just passed untouched to the message queue. They are not used by the algorithm that generates the next event. The event(underscore)master table contains the following columns relative to work flow generation:

\* event(underscore)id - The key to the table.

\* enabled - A flag indicating whether the event is enabled or not. If the event is not enabled, no ToDos are generated when the event occurs.

\* col(underscore)id(underscore)1 to col(underscore)id(underscore)32 - The column master column ids of the column values generated by the event. The next(underscore)step table contains the following columns relative to work flow generation:

\* msg(underscore)id - A unique message id. Every workflow event description has a unique message id.

\* event(underscore)id - The event id. The same event(underscore)id may have multiple next(underscore)step's defined in this table with different msg(underscore)id's.

\* enabled - A flag that indicates if the message is enabled or not. If the message is not enabled no ToDo for this msg(underscore)id is generated when the event occurs.

\* activity(underscore)id - The activity(underscore)id of the next activity in the workflow to be generated in response to the event for this next(underscore)step.

\* no(underscore)dup(underscore)msg - A flag which indicates whether duplicate messages are allowed. If this flag is set then a ToDo will not be generated if a ToDo already exists with the same owner, activity(underscore)id, msg(underscore)id, and column values. The next(underscore)step(underscore)options table contains the following columns relative to workflow generation:

\* msg(underscore)id - The msg(underscore)id of the next(underscore)step(underscore)options. The msg(underscore)id and the next(underscore)step(underscore)ent(underscore)val form the key to this table.

\* next(underscore)step(underscore)ent(underscore)val - The value that is matched to the @p(underscore)next(underscore)step(underscore)ent(underscore)val parameter passed into the

stored procedure to determine the options for this event. If no row exists where @p(underscore)next(underscore)step(underscore)ent(underscore)val equals this column, then row where next(underscore)step(underscore)ent(underscore)val is "\*" is used if it exists.

\* enabled - A flag that indicates if the message is enabled or not. If the message is not enabled no ToDo for this msg(underscore)id is generated when the event occurs.

\* user(underscore)id - The user(underscore)id that the message is assigned to if assign(underscore)to is "U".

\* msg(underscore)group(underscore)id - The Workgroup ID of the workgroup the message is assigned to if assign(underscore)to is "G" or if assign(underscore)to is "D" and the @p(underscore)assign(underscore)to(underscore)arg is "D".

\* track(underscore)hist - If set then the ToDo is copied to the message(underscore)queue(underscore)hist(underscore)l table when the ToDo is deleted.

\* assign(underscore)to - Parameter which indicates who the event should be assigned to as follows:

\* "S" - Assign to the user who generated the event.

\* "D" - Assign to the user or workgroup specified by the @p(underscore)assign(underscore)to(underscore)arg and @p(underscore)to(underscore)owner(underscore)arg. If @p(underscore)to(underscore)owner(underscore)arg is "D", assign to the group specified by the msg(underscore)group(underscore)id column.

\* "U" - Assign to the user specified by the user(underscore)id column.

\* "G" - Assign to the group specified by the msg(underscore)group(underscore)id column.

In summary, therefore, the work flow of the previously described embodiment of the present invention operates as follows:

\* An application generates an event (220 in FIG. 2).

\* A single event can trigger multiple NextSteps (230), each of which can trigger a specific activity (250). The activity is fixed by the workflow definition in the next(underscore)step table..

\* For each NextStep 230 the next owner of the message is determined by the NextStepOptions. The NextStepOptions allow conditional generation of the owner of the ToDo based on equality of a column in next(underscore)step(underscore)options table and a single parameter passed as part of the event.

The previously described work flow engine is basically stateless. When a user 120 generates an event, the work flow engine behaves the same in response to the event independent of how the user 120 initiated the activity and independent of any workflow events that may have occurred in prior steps.

### Conditional Workflow

In order to further describe how the conditional work flow feature of the present invention may be implemented, the following terms and concepts are defined below.

**Conditional Logic Types.** Conditional logic can be used to determine the next step in the work flow process (e.g., elements 200 and 230 of FIG. 2), to determine to whom the next step should be assigned (e.g., element 240 of FIG. 2), or to select which approvers on an approval list should be used. In one embodiment, the following operations may be used as conditional logic:

- \* Comparing an integer or numeric column to a constant. The equal, not equal, less than, less than or equal to, greater than, or greater than or equal to operations may be supported.
- \* Comparing a string column to a constant. The equal to, starts with, and contains contains operations may be supported.
- \* Boolean logic which connects several comparative expressions. These operators may include AND, OR, and NOT. The present workflow engine may allow the workflow designer to choose any field that the window passes to the trigger event function (element 220 of FIG. 2) within conditional logic. An enhanced trigger event function may be implemented to allow applications to pass more than 16 values.

**Approval Lists.** Adding approval lists to the present invention adds the following functionality:

- \* Provide approve and reject as activity actions. This allows the approver to see the object that they are approving. Otherwise, approvers in a work flow use an approval window and have to zoom to the actual object.
- \* Provide a consistent handling of approvals. This ensures that all the approval windows have the same look and feel. It also decreases the development cost of adding approvals to new windows and the maintenance cost of the approval maintenance windows.
- \* Automatic support for new Approval features as they are added to the platform. These include substitutes, roles, and conditional generation.
- \* Allow users to approve items directly from the Task Details list (element 5601 of FIG. 56) without actually entering the application activity.

**Structures Integration.** The present invention may be implemented to support three structures based functions that can be used within workflow definitions. Structures corresponds to heirarchical structured data in a database, such as that built into the SmartStream family of

products available from Dun & Bradstreet Software Services, Inc., and is described in further detail in the co-pending U.S. Patent Application filed on May 26, 1995, and entitled Method and Apparatus for Protecting the Integrity of Structured Data (serial number not yet assigned), which is incorporated herein by reference.

The structures-based functions that may be supported by the present invention are: IsIn(), UpOne(), and UpToLayer(), as described in further detail below.

Three Structures functions are available for work flow assignments. Point, Immediate Ancestor and Ancestor at Layer. To support these functions the Structures product (available from Dun & Bradstreet Software Services, Inc.) is enhanced to allow a work flow assignee to be associated with a point in the database structure.

The Point function takes as input a structure group, a structure name and a column that is used as a point name. It determines the assignee by getting the work flow assignee at the point given.

The Immediate Ancestor function takes as input a structure group, a structure name and a column that is used as a point name. It determines the assignee by getting the work flow assignee at the point which is the immediate ancestor of the point specified.

The Ancestor at Layer function takes as input a structure group, a structure name, a layer name and a column that is used as a point name. It determines the assignee by getting the work flow assignee at the point which is the first ancestor of the point specified at the indicated layer.

Next Step Processing. In the implementation of the present invention described previously, next step processing requires that the initial values to the next step be provided in the same order as the keys passed to the pamp004(underscore)next(underscore)step function. In a further embodiment, the next step processing may be changed so the keys from the activity can be provided in any order relative to the keys in the To Do List.

Step Completion. In the embodiment of the present invention described previously, a step is completed when the user enters an activity from the To Do List and subsequently performs a save or a delete within the activity. In a further embodiment described below, the step will be completed only if the user subsequently performs a save or a delete with the same keys as were passed in with the To Do.

Conditional Next Step Logic. The workflow engine described previously supports no conditional generation of the next step activity (230 in FIG. 2). In a further embodiment, the workflow engine may allow simple conditional logic based on any column value passed to the trigger event function. Each next step definition (230) will define the conditions for which it is applicable.

Conditional Owner Logic. In the embodiment of the present invention described previously, the workflow engine supports limited conditional generation of the owner based on the next step entity. In a further embodiment of the present invention, the workflow engine will allow conditional assignment based on all the comparison operators using all the column values as data.

**Data Values.** With the workflow design described previously, the system allows for 16 data values to be passed on to the next step in the workflow. In a further embodiment, the trigger event function will accept up to 32 column values. These values can be used for conditional logic or can be passed on to the next step. Within event(underscore)master (see FIG. 14D), a flag may be added to indicate whether or not a column value is a key column.

The design standards may also be changed to recommend that as many values as is possible be included in each call to trigger event. For windows where more than 32 columns are available, the developer preferably should pick the columns that are most likely to be passed onto other windows or used in workflow conditions for inclusion in the trigger event parameters. Any column value may be passed to the trigger event function even if the value is not a key or does not actually appear on the window.

Existing application windows may continue to work unchanged; however, only those values that are currently explicitly passed may be available for use in workflow conditions. Application developers are encouraged to increase the number of column values provided to the trigger event function for existing windows.

**Window Initial Values.** The present invention may be implemented to require that application windows that participate in workflows provide a dictionary of what columns each activity can accept as input. The workflow engine may use this dictionary to take care of re-ordering column values so that activities can be connected and parameter ordering is not important. The dictionary may also be used by the workflow workbench user interface to make it easier for the designer to connect the data between activities.

The workflow engine may allow values to be remapped, so that a field with one column id from the source window can be mapped to a different field with a different column id on the destination window.

In order to always save the key values for tracking workflows, the present invention may be designed to require that the columns in event master be marked as key values or non-key values. The key values for an event are the values that uniquely specify the instance of an event. In general these values will be the same as the key values for the window; however, there is no restriction that this must be the case.

**Administrative Interface.** The present invention may be designed to provide a completely new graphical user interface for defining and describing workflows. This user interface may replace the previous workflow workbench with a new user interface that is more intuitive and which supports specification of conditional next steps, conditional assignments, and approvals.

**Mail Integration.** The present invention may be designed to support mail integration by creating an "attachment". Mail would be used as the delivery mechanism for To Do messages, but would use the workflow system of the present invention as the "forms package".

When a user of the present invention receives an E-mail To Do and clicks on the attachment portion, the system would check to see if the present invention was operational. If so, the system

would simply send a DDE (Dynamic Data Exchange) message to start the appropriate activity window to process the ToDo. If the present invention was not active, it would be started which would bring up the login box followed by the activity window to process the To Do. When the user terminated the activity window, the present invention would remain operational. FIG. 57 shows a mail-enabled To Do in a users inbox.

**Roles.** The present invention may be implemented to provide the ability to define roles within a company. The workflow designer could then define workflow owners in terms of roles. The workflow engine resolves roles at runtime to determine the real user or group. This functionality allows for easier re-programming when a user leaves the company or changes jobs.

**Substitutes.** The present invention may be implemented to allow the user or administrator to define a substitute for a user. When a user has a substitute defined for them, all the user's To Do's are sent to the substitute rather than to the specified user. A substitute can only be defined for a user -- it cannot be defined for a workgroup or a role.

FIG. 11 depicts a functional block diagram of the present invention adapted to implement the features described above. The basic structure of the workflow design that may be utilized with the system of FIG. 11 is similar to that described with respect to FIGS. 1A, 1E, 1F, 1H and 2-52. However, for clarity, new figure reference numerals are used in FIG. 11 -- of course, it will be readily understood that certain components in FIG. 11 generally correspond to like components in FIGS. 1A, 1E, 1F, 1H and 2-52.

Referring to FIG. 11, the workflow administrator 10001 may use a Workflow Workbench 10005 to define the workflow for the present invention. The Workflow Workbench uses the Workflow Meta Data 10010 (activity(underscore)master, event(underscore)master, etc.) to show what options are available. The workflow definitions created by the workbench 10005 are stored in the database in the Workflow Definition tables 10015 (next(underscore)step, next(underscore)step(underscore)options, etc.).

When a user 10002 performs an action in an activity window 10020 that modifies application data, the application activity uses Application Architecture functions 10025 to call the Workflow API 10030 to trigger an Event 10035 while passing in the relevant data values 10040. The Workflow API 10030 calls the Workflow Engine 10045 which uses the event 10035, the data 10040, the Workflow Definitions 10015, and the Workflow Meta Data 10010 to generate a next step. The next step and the data associated with it are stored on the Workflow Message Queue 10050. The user may see next steps as Tasks on a To Do list 10055. When a user processes a To Do message in the To Do list 10055, the message is deleted or marked as complete on the Message Queue 10050. Additionally if the system is set up to track history a copy of the message is moved to the Workflow Tracking tables 10060 (message queue history).

Applications 10065 not designed for use with the present invention can interface with the workflow system via the Workflow API 10030. The Workflow API 10030 allows such applications 10065 to trigger events 10035, read the To Do's 10055 that are on the message queue 10050, and change the status of To Do's on the message queues 10050.



The Workflow Workbench 10005 may be designed to provide an improved user interface and to support conditional logic and structures integration. The new features being provided may require additional Workflow Meta Data tables 10010 and changes to the existing Workflow Meta Data tables 10010. The Workflow Definitions tables 10015 may also be enhanced to support conditional logic and structures integration.

The Workflow API 10030 may be modified to support new workflow functions and to track changes in the Workflow Definition 10015 and Message Queue tables 10050. The application architecture functions 10025 which interface with the Workflow API 10030 may be designed to be backwards compatible; however, in this case it is advantageous that applications be implemented to provide more data values to these functions.

The Workflow Engine 10045 may be enhanced to resolve conditional logic (described elsewhere) including heirarchical database Structures integration and to deal with the table structures in the Workflow Meta Data 10010, Workflow Definition 10015, and Message Queue tables 10050. The Workflow Engine 10045 may also include logic to resolve roles and substitutes, defined previously.

Approvals may be built into the workflow. Approvals build on the application specific approval processes that are already in place, and require the application developer to do the following:

- 1) Add an approval table in the application database. The key of approval table is the application table key, a sequence number, and an owner. The entries with the key matching the key of the application table row would be the approval list for that row.
- 2) The platform provides approval lists. When an approval is the next step the approval list to be used is passed to the approval window as data. Conditional logic can be supported in the determining of which approval list is to be used. Dynamic generation of the approval list will allow the name of the approval list to be used to be a field on the application window.
- 3) The platform will also provide a generalized window for tracking approvals. This window will be the management interface into the application provided approval table. The approval tracking window may either be an ancestor window that the applications use to create their own descendant tracking window with the proper keys or may dynamically modify itself touse the proper keys and data.

The following table (Table M) illustrates those database tables that may be changed from those tables previously described in order to implement the conditional workflow features of the present invention. The first column describes the table name, the second column describes the family of the table, and the third column summarizes the change to the table. The following database tables shown in Table N may be added to support conditional workflow. In this case, the third column describes the purpose of the table.

FIG. 54 illustrates an E/R diagram for the tables that have been changed or added to support conditional workflow. This diagram is described in further detail below.

The database tables summarized in the tables above that are either changed or created are described in further detail below. Some of these tables are described in further detail in the appendices to U.S. Patent Application Serial No. 08/213,022, filed March 14, 1994, and U.S. Patent Application Serial No. 08/475,575, filed June 7, 1995, both of which are incorporated herein by reference thereto.

`next(underscore)step`. The next step table is modified with the addition of `forward(underscore)allowed` flag. If this flag is set for a To Do then the target user can forward the To Do to another user.

`next(underscore)step(underscore)options`. Conditions may be added to each Next Step Option. Thus an entity `(underscore)seq(underscore)num` field will be added to each option as part of a key. The options that have the same `msg(underscore)id` and `next(underscore)step(underscore)entity` but with different `entity(underscore)seq(underscore)num` values represent different conditions for the same entity. Since the workflow engine must always come up with an assignment for every To Do, the last conditions must represent the "else" condition. The workflow workbench GUI will ensure that this is true.

The assignee in `next(underscore)step(underscore)options` may be a structure function. To support structure functions, the following columns are added to this table which are used when the `assign(underscore)to` field indicates a structures function:

- \* `struct(underscore)func(underscore)type` - Indicates which function (UpOne or UpToLayer) is to be performed.
- \* `struct(underscore)group` - The name of the structure group that the structure is a member of.
- \* `structure(underscore)name` - Indicates the name of the structure that is to be traversed.
- \* `col(underscore)id` - Indicates the column value that is used to find the starting point in the structure.
- \* `default(underscore)type` - Indicates whether the user or workgroup should get the message if the structure function fails.
- \* `layer(underscore)name` - Holds the layer name parameter to the function if the structure function is UpToLayer(). `message(underscore)queue(underscore)l`. The following changes will be made to the message queue:
  - \* The key will be changed to be `server(underscore)name` and `seq(underscore)num`. The `server(underscore)name` will be the name of the server where the message was created. The `seq(underscore)num` column will be a number which uniquely identifies a message within a server. Both values are required as part of the key so that every message has a system-wide unique key for tracking purposes. If a message is reassigned to a user or workgroup on a different server, then `server(underscore)name` column will be the server where the message was first created not the name of the server where the message is stored.

\* The create(underscore)time, owner(underscore)id, and owner(underscore)type fields will become non-key columns in this table.

\* The message queue will be normalized thus the col(underscore)id(underscore)x and col(underscore)val(underscore)x columns from the message(underscore)queue table will be moved to the message(underscore)queue(underscore)columns table. The key to this table will be server(underscore)name, seq(underscore)num, and col(underscore)id.

\* The prev(underscore)server(underscore)num, prev(underscore)seq(underscore)num, and last(underscore)insert(underscore)type columns are added to support enhanced tracking. If the message was inserted when not processing a To Do, because a To Do was reassigned, or because a user was moved from one server to another, then the prev(underscore)server(underscore)num and prev(underscore)seq(underscore)num indicate the key of the previous message. The last(underscore)insert(underscore)type will indicate the type of the insert. The type of the insert can be either no previous message, next step, reassign, or user movement. message(underscore)queue(underscore)hist(underscore)1. The message queue history table may be changed to match the message queue table. Thus the following changes will be made:

\* The key will be changed to be server(underscore)name and seq(underscore)num. The server(underscore)name will be the name of the server where the message lived when it was processed. The seq(underscore)num field will be a numeric datatype.

\* The owner(underscore)id, owner(underscore)type, activity(underscore)id, msg(underscore)id, and delete(underscore)time fields will become non-key columns in this table.

\* The message queue history table will be normalized thus the col(underscore)id(underscore)x and col(underscore)val(underscore)x columns will be moved to the message(underscore)queue(underscore)hist(underscore)columns table.

\* The prev(underscore)server(underscore)num, prev(underscore)seq(underscore)num, and last(underscore)insert(underscore)type columns are added to support enhanced tracking. If the message was inserted when not processing a To Do, because a To Do was reassigned, or because a user was moved from one server to another, then the prev(underscore)server(underscore)num and prev(underscore)seq(underscore)num indicate the key of the previous message. The last(underscore)insert(underscore)type will indicate the type of the insert. The type of the insert can be either no previous message, next step, reassign, or user movement. event(underscore)master. The event master table will be normalized -- thus the col(underscore)id(underscore)x columns will be moved to the event(underscore)master(underscore)columns table.

next(underscore)step(underscore)conditions. The next step conditions table will contain the conditional expression associated with a next step. The rows in the next step conditions table with the same msg(underscore)id contain the expression definition in postfix format. This table contains the following columns:

\* msg(underscore)id (primary key) - The msg(underscore)id of the next step the condition is associated with.

\* express(underscore)seq(underscore)num (primary key) - The sequence number of the row within the expression. This indicates the order of the operands in postfix notation.

\* operator - This indicates the postfix operator. This can be an arithmetic operator (=, >, <, <=, >=, <!, >!, or <!=), a string operator (equals, starts with, or contains), the function operator, or the push operator. If the operator is an arithmetic or string operator the function is performed on the two operands on the stack. If the operator is the function operator, the parm(underscore)type and parm columns indicate the function definition and the function is performed on the values on the stack. The result of the function is then pushed onto the stack. If the operator is the push operator, the parm(underscore)type and parm parameters indicate the constant or column value that is to be pushed onto the stack.

\* parm(underscore)type - If the operator is push this column indicates what value is to be pushed. It may indicate a constant is to be pushed, a column is to be pushed, the user id is to be pushed or the next step entity is to be pushed.

\* parm - If parm(underscore)type is constant, this is the value to be pushed. If the parm(underscore)type is a column, this contains the column id. If the operator is function, this column contains the function name. next(underscore)step(underscore)parameters. The next step parameters table contains a source activity to destination activity mapping of columns and indicates which columns are displayed on the Task Details GUI described elsewhere. It contains the following columns:

\* msg(underscore)id (primary key) - The msg(underscore)id of the next step the mapping is associated with.

\* col(underscore)id (primary key) - The col(underscore)id column contains the column id of the column provided by the event.

\* dest(underscore)col(underscore)num - The index of the column in the init(underscore)values() array that is passed to the target activity. If the column is displayed but not passed to the target window then this value is 0.

\* dest(underscore)col(underscore)id - The column id of the value from in the destination window.

\* display(underscore)num - This value indicates the column where the value is displayed on the Task Details GUI. next(underscore)step(underscore)options(underscore)conditions. The next step options table contains the conditional expression associated with next step options. The key is msg(underscore)id, next(underscore)step(underscore)ent(underscore)val, entity(underscore)seq(underscore)num, and expression(underscore)seq(underscore)num and the non-key columns are the same as in the next(underscore)step(underscore)conditions table (operator, parm(underscore)type, and parm).

message(underscore)queue(underscore)values. The message(underscore)queue(underscore)values table contains the column values that are passed along with the ToDo. These values may include values that are keys to the triggering event, values that are passed on to the target activity, and values that are displayed on the task detail GUI (see FIG. 16L). This table contains the following columns:

- \* server(underscore)number (primary(underscore)key) - The server number where the message was generated.
- \* seq(underscore)number (primary key) - The sequence number of the To Do.
- \* col(underscore)id (col(underscore)id) - The column id of the value as it is passed in the event.
- \* col(underscore)val - The value of the column.
- \* key(underscore)num - If the column is a key value to the event, then this column contains the number of the key. If the column is not a key value, this column contains 0.
- \* dest(underscore)col(underscore)num - If the column is an initial value for the target activity, then this column contains the index of the column in the init values array. If the column is not an initial value, then this column contains 0.
- \* dest(underscore)col(underscore)id - This column contains the column id of the column in the target activity init(underscore)key(underscore)i array.
- \* display(underscore)num - This column contains the column where the value is displayed in the Task Details GUI. If the column is not displayed in the Task Details GUI, then this column contains 0. message(underscore)queue(underscore)hist(underscore)values. The message(underscore)queue(underscore)hist(underscore)values contains the same columns as the message(underscore)queue(underscore)values table.

activity(underscore)input(underscore)values. The activity input values table contains a list of the column ids that an activity can take as input. This table contains the following columns:

- \* activity(underscore)id (primary key) - The activity id of the activity.
- \* dest(underscore)col(underscore)num (primary key) - The number of the parameter in the init(underscore)values array.
- \* dest(underscore)col(underscore)id (primary key) - The column id of the parameter.
- \* key(underscore)flag - A boolean flag indicating whether the column is a key column or not. For a particular dest(underscore)col(underscore)num, an activity can accept column ids as input. This feature allows "column synonyms", although they must be separately specified for each window in the system.

roles. The roles table contains a list of roles and owner names. It contains the following columns:

- \* `role(underscore)name` (primary key) - The name of the role.
- \* `role(underscore)seq(underscore)num` (primary key) - The sequence number of the conditional for the role. Conditions are evaluated in sequence number order and the first condition that is true indicates the value of the role.
- \* `owner(underscore)type` - The type of the role. This can be either a workgroup, a user, or a function.
- \* `owner` - The workgroup, user, or function name the role should map to. The role table contains support for conditional role generation even though this feature will not be provided until a future release.

`role(underscore)conditions`. The role conditions table contains the conditions associated with a role. It contains the following columns `role(underscore)name`, `role(underscore)seq(underscore)num`, and `expression(underscore)seq(underscore)num` columns as keys and the operator, `parm(underscore)type`, and `parm` columns as non-key values. The meaning of these columns in this table is the same as the meanings of these columns in the `next(underscore)step(underscore)conditions` or `next(underscore)step(underscore)options(underscore)conditions` tables.

`event(underscore)columns`. The event columns table contains the column identifiers for an event. It contains the following columns:

- \* `event(underscore)id` (primary key) - The event identifier.
- \* `column(underscore)num` (primary key) - The number of the column when passed to the trigger event function or stored procedure.
- \* `col(underscore)id` - The column identifier of the column value.
- \* `key(underscore)num` - Indicates what number key the column is for the event. If the column is not a key value then this field contains 0. `substitutes` tables contains a list of users who have substitutes defined for them. It contains the following columns:
  - \* `user(underscore)id` - The `user(underscore)id` of the user whose messages will be sent to the substitute.
  - \* `subst(underscore)owner` - The user id or workgroup id of the substitute user or workgroup.
  - \* `subst(underscore)owner(underscore)type` - Indicates whether `subst(underscore)owner` is a user or workgroup. `approval(underscore)list`. The approval list table contains the header information for each approval list. It contains the following columns:

- \* `apprvl(underscore)list(underscore)id` - The name of the approval list.
- \* `apprvl(underscore)activity(underscore)id` - The activity the approval list is defined for.
- \* `notes` - Notes that the user can use to describe an approval list.
- `approval(underscore)list(underscore)detail`. The approval list detail table contains a line item for each approver on the list. It contains the following columns:
  - \* `apprvl(underscore)list(underscore)id` - The name of the approval list.
  - \* `apprvl(underscore)activity(underscore)id` - The activity the approval list is defined for.
  - \* `apprvl(underscore)level` - The level for this approver on the list.
  - \* `apprvl(underscore)id` - The user ID, workgroup or role of the approver.
  - \* `apprvl(underscore)type(underscore)code` - The type of `apprvl(underscore)id`, either user (u), workgroup (g), role (r), or structure function (f).
  - \* `structure(underscore)function(underscore)type` - The type of the structure function if `apprvl(underscore)type(underscore)code` is (f).
  - \* `structure(underscore)group(underscore)id` - The structure group ID used for the structure function if `apprvl(underscore)type(underscore)code` is (f).
  - \* `structure(underscore)name` - The structure name used for the structure function if `apprvl(underscore)type(underscore)code` is (f).
  - \* `col(underscore)id` - The column ID of the column used as the point name input if `apprvl(underscore)type(underscore)code` is (f).
  - \* `layer(underscore)name` - The layer name used for the structure function if `approval(underscore)type(underscore)code` is (f) and the `structure(underscore)function(underscore)id` indicates Ancestor at Layer.
  - \* `default(underscore)type` - The type of the default assignee to be used when `apprvl(underscore)type(underscore)code` is (f) and the structure function fails for some reason.

Almost all the workflow stored procedures and workflow API functions described previously for this invention will preferably require changes in order to implement the conditional logic embodiment of the present invention because of the change in the message queue key, the increase in the number of column values passed in to 32, the increase in the potential size of all column values to 255 bytes, and the normalization of the message queue, message queue history, and event master tables. The following tables list each stored procedure and workflow API function, indicate whether the external interface to the functions requires changes, and describes what any external or internal changes are necessary.

To minimize the impact of future changes the message queue key will preferably be returned to the workflow API as a 100-byte character string. Applications which use the workflow API should therefore not create any dependencies on the internal format of this string.

The following table (Table O) describes the interface changes necessary for the Workflow API functions of the previously described invention (e.g., SmartStream 3.0 available from Dun & Bradstreet Software Services, Inc.).

The following table (Table P) describes the interface changes necessary for the stored procedure functions of the previously described invention.

Several footnotes should be considered when reading the above tables. The numbers to the left of the footnotes correspond to same numbers in the above tables:

(1) These stored procedures only deal with 10 columns in a message; therefore, they appear to be obsolete. Thus, they should preferably be deleted and the windows for which this causes problems should be fixed accordingly.

(2) These stored procedures are no longer supportable with the distributed work flow feature of the present invention.

(3) These stored procedures are still used by the workflow API; however, due to changes to the way the distributed nature of the present invention is handled, applications should preferably not call them directly.

(4) Callers of these stored procedures must make sure they are operating against the correct `message(underscore)queue`.

The trigger event stored procedure will additionally require changes to evaluate conditional logic. These changes are described in the following section.

`psp(underscore)trigger(underscore)ams(underscore)event` Stored Procedure. The following changes will be made to the `psp(underscore)trigger(underscore)ams(underscore)event` stored procedure:

**Activity Input Table** - The activity input values table will be used to reduce and order the column values that are included in the generated message.

**Next Step condition evaluation** - Each next step definition has a conditional attached to it. The next step is only triggered if the condition is true. The workflow engine will have to evaluate the condition to determine if the `ToDo` should be generated.

**Assignee condition evaluation** - The Assignee (next step options) of each workflow event will be able to have conditional logic attached to it. The workflow engine will have to evaluate the conditional logic to determine who the Assignee is.



Role resolution - An assignee can now be a role. When an assignee is a role the workflow engine will have to resolve the role.

Substitute processing - If the user has a substitute defined assign change the assignee to the substitute user.

Approvals -- Approvals are described in further detail in a later section. The new algorithm for the psp(underscore)trigger(underscore)ams(underscore)event stored procedure is described below in conjunction with FIG. 55 as follows: Both next step processing and next owner processing require that conditional expressions be evaluated. One expression evaluation stored procedure may be written to handle all expression evaluation. The trigger event function will put the column values into one temporary table and the expression to be evaluated into another temporary table. The expression evaluation stored procedure will evaluate the expression using these two temporary tables and return a boolean TRUE or FALSE to indicate the value of the expression.

Conditional workflow will preferably require the following application architecture changes:

Approve, Reject. Approve and Reject are few actions that are only available for windows that support approvals. Approve and Reject will execute the logic that is currently in the approval windows. These actions are described in further detail in a later section.

Fix step completion bugs. The logic within basic window which controls when a To Do is marked as complete will be fixed so that it only marks the To Do complete when a save or delete is done with the same keys as the keys provided in the initial data coming into the window. This requires resetting the AM(underscore)next(underscore)msg(underscore)flg(underscore)i basic window instance variable when the window key values change due to the user typing a new key value.

The administration interface for the present invention may include the following new or modified windows:

Work Flow Workbench. A new graphical-based interface may be implemented for defining next steps and next step options (owner assignment). The graphical interface will preferably create conditional workflow definitions required by the new table layout. This includes compiling any conditional logic expressions into postfix notation before storing them in the appropriate conditional table. This work flow workbench mapping tool is defined in further detail in a later section.

Approval Tracking. This window may be keyed by the application key. It may display the current status of the approval process for the item shown. It also shows a list of the future approvers. Approvals are described in further detail in a later section.

Approval List Definition. This window allows the administrator to create an approval list. Each list is keyed by the list name and the activity class for which it is applicable. The global activity

class (\*) means that the approval list can be used for any window in the system that supports approvals. Approvals are described in further detail in a later section.

**Role Definition.** The Role Definition window is a simple tabular window used to maintain the role table. The key for this window is the role name. The data fields are the owner type (user or workgroup) and the owner.

**User Substitute Definition.** The Substitute Definition window is a singlerow window that allows a user to define a substitute for himself. The key will always be the user's user(underscore)id and will not be able to be modified. The window will display assign(underscore)to type and the user or workgroup.

**Administrator Substitute.** The Administrator Substitute window is a simple tabular window that allows the administrator to change any users substitute. The key is user(underscore)id and the data values are assign(underscore)to type and user or workgroup.

**Activity Input Data.** The Activity Input Data window is a keyed tabular window used to maintain the activity(underscore)input(underscore)data table. The key to the top section is the activity id. The fields shown in the tabular and singlerow portions of the window are parameter number, column id, column description, and key flag.

**Events.** The event window needs to be changed to support 32 columns within an event. This can either be done by modifying the current window or by changing the window to be a basic tabular.

Applications will have to make the following changes to work properly with the conditional logic features of the present invention:

- 1) Change any initial data loads that load data into event(underscore)master to load data into both event(underscore)master and event(underscore)columns.
- 2) Change any initial data loads for the next(underscore)step table to load data for the new columns in this table.
- 3) Change any initial data loads for the next(underscore)step(underscore)options table to load data for the new columns in this table.
- 4) Create initial data for the activity(underscore)input table to describe the values that an activity can accept as input.
- 5) Any application that uses the workflow API or which directly calls any workflow stored procedues in the wact or wjt table families will have to be changed to use the new workflow API functions.

Applications will also have to make the following change to allow userss to fully exploit the capabilities of conditional workflow: Change all calls to trigger event functions to include all

column values that may be relevant for passing on to other activities or for use in conditional logic.

In order for the present invention to work with Structures, Structures integration requires that all structures that are used within workflows be replicated to all work flow servers in the system. Furthermore, the use of the structure functions as the assign to in next step options requires that structures be enhanced to include work flow assignee fields within the point definition.

## Approvals

### Definitions

Approval Enabled Activity - A window that supports user configuration of Approval workflow by calling the new Approval trig(underscore)event function and allowing for the possibility that no Approval process was required. The activity is identified to the workflow mapping tool through the new apprvl(underscore)enabled(underscore)activity table.

Approval Enabled Event - An event that allows users to map a workflow using an Approval List for distribution of Next Step messages.

Approval List - A list of Users, Workgroups, Roles or structures functions which supports sequential grouping used in an Approval process for an Approval Enabled Activity.

Approval Tracking/Status - An application window/table that supports tracking of a single instance of the Approval process and access to Approval comments. Shows Approval level, approver, name and type, Approval Status.

Substitutes - A User or Workgroup that will temporarily receive Workflow messages intended for a specified User. This will apply to ALL Workflow not just Approvals.

Workflow Mapping Tool - A completely new graphical user interface for defining and describing workflows. It is intuitive and supports specification of conditional Next Steps, conditional assignments and assignment of Approval Lists. It is described in further detail below.

The main goal of the Approvals portion of the present invention is to provide the user 120 with a way to customize the Approval process with the least amount of work required by the applications. A secondary goal is to provide a model, with supporting tools, to standardize the way we do Approvals. This will allow automatic support of new Approval features as they are added to the platform. These include Substitutes, Roles, and conditional generation

Four application windows which use an Approval process have been used as the basis for the design. Currently, each transaction window that supports approvals has:

- 1) An Approval List table, stored procedures and a maintenance window with a "Copy From" response window.

- 2) An Approval Substitutes table, stored procedures and a maintenance window.
- 3) An Approval Tracking/Status table to monitor the approval process of a specific instance, a comments table, stored procedures and a maintenance window.
- 4) Some mechanism for either approving or rejecting and handling the updates to tracking/status tables.
- 5) Scripts to read the instance table, generate the appropriate To Do messages and to recognize when the Approval process is complete.

Payment Request and Journal Entry Approvals are very similar in that they both use the common Approval List/Copy From and Substitutes ancestor windows and have an Approval Status table with associated Comments. Purchase Requisitions use a similar format for Lists and Substitutes but have these tables in a different table family than the Status and Comments tables. They also support Approval at both the line and the document level. All three use the Approval Tracking window as the Next Step activity in the Approval process, thus making the object being approved only visible at Approval time through Zoom. Requisitions provides an Approval view on the Requisition maintenance window. ECR/ECN seems the most different from the others with Comments associated with the document being approved rather than the Status table, Approval from the document window and different column names and datatypes for Lists and Substitutes.

To provide Approval Enabled activities, i.e. activities which support user modification of the Approval process through the use of the Workflow Mapping Tool, and to integrate new workflow features in to that process, the following tasks must be accomplished:

- 1.) Provide a mechanism to identify Approval Enabled activities and events and their components, e.g. the Approving activity. Also, indicate when a To Do is for an Approval activity.
- 2) Consolidate multiple application Approval List tables into a single platform table. Add support for structures based functions that can be used within workflow to determine workflow assignees. Structures which have the workflow global dynamic property can be used to determine the assignee within a workflow. Two such functions are provided:

Determine the assignee by reading the workflow dynamic properties of the direct ancestor of a point.

Determine the assignee by getting the workflow dynamic properties of the ancestor of a point at a given layer.

Determine the assignee by getting the workflow dynamic properties of a point.

- 3) Incorporate Approval List Substitutes into the general workflow Substitutes design. When a user has a substitute defined for them, all the user's To Do's are sent to the substitute. A

substitute can only be defined for a user -- it cannot be defined for a workgroup or a role. When a substitute is defined, To Dos that are already in the users To Do list are not affected

4) Provide a model window for consistent handling of Approval Status/Tracking. This would make sure that all the Approval Tracking windows have the same look and feel. It would also decrease the development cost of adding Approvals to new windows. The existing application tracking windows can still be used. The GUI is generally consistent.

5) Provide a PowerBuilder object/ancestor to support Approve, Reject and Comments from within an application window which will allow a view of the object being approved. This change would allow the approver to see the object that they are approving. Currently, approvers use the Approval Tracking window and have to Zoom to the actual object. For this release, the view of the actual object will be read-only.

6) Allow users to Approve items directly from the Task Details list without actually entering the application activity.

7). Provide workflow functions which will consolidate code so that an Approval Enabled window can use two functions to do ALL of the work required including

Generating rows for the tracking/status instance table for each approver on the user selected approval list

Generating To Dos for the first level of approvers

Generating any additional To Dos for Next Steps attached to the Approval enabled event

Returning the ID and the type of the approver used or an indicator that no approval process was required

New workflow functions may be written to support Approvals. Both are a variation of `pam0011(underscore)trig(underscore)event()`. The first, `pam0012(underscore)trig(underscore)event()`, will return a code indicating Approval Complete and will require three (3) additional arguments, `tracking(underscore)SQL(underscore)string`, `approver(underscore)used` and `approver(underscore)type(underscore)used`.

The flow of the new function will be:

- \* Do the same things that the old `trig(underscore)event` function does PLUS

- \* If the approval flag is set in a `next(underscore)step` then evaluate the `next(underscore)step(underscore)options` conditions to get the Approver ID and `approver(underscore)type(underscore)code`.

- \* If no approval is required, i.e. no list value is present, return 1, an indicator that the Approval is complete.

\* Otherwise...

Update the `approver(underscore)used` and `approver(underscore)type(underscore)code` arguments for return to the caller .

\* Using the application window provided string containing `svr(underscore)db(underscore)owner.stored(underscore)proc` and the formatted key values, insert rows into the Tracking table. Resolve Structures functions to provide the User/Workgroup/Role ID and the Approval Level for each approver. Set the status to N.

The second, `pam0013(underscore)notify(underscore)approvers()`, will generate the next batch of Approval To Dos as specified by the `apprvl(underscore)level` and will require one (1) additional argument, `next(underscore)approver(underscore)SQL(underscore)string`. It can also be used in the Approval window.

With the prior system, when an approval process is associated with an activity:

\* From `dbupdate` event, call `update(underscore)insert(underscore)tran()` to insert a row into the database for the object to be approved.

\* error handling

call `pam0011(underscore)trig(underscore)am(underscore)event(insertevent,... )`

select the owners from the approval list

build a `SQL(underscore)string` for each approver for the insert stored procedure for the instance table

insert a row into the instance table for each approver

call `pam0011(underscore)trig(underscore)am(underscore)event(sendapprmsg,...)` for each approver in the first batch.

\* Using the new functions, call `update(underscore)insert(underscore)tran()` to insert a row into the database for the object to be approved

error handling

`complete(underscore)var = pam0012(underscore)trig(underscore)approval(underscore)event(insertevent .....SQL(underscore)string, approver(underscore)used, approver(underscore)type(underscore)code` where `SQL(underscore)string = am(underscore)server(underscore)db(underscore)owner(underscore)g + insert(underscore)instance(underscore)stored(underscore)procedure + & instance(underscore)keys` (formatted for SQL)

pam0013(underscore)notify(underscore)approvers(sendmsgappr, .., .., SQL(underscore)string, approval(underscore)level) where SQL(underscore)string = am(underscore)server(underscore)db(underscore)owner(underscore)g + date(underscore)instance(underscore)stored(underscore)procedure + & instance-keys (formatted for SQL) Note: The stored procedures should be written such that the values for the tracking/status table are the last parameters passed in.

The description below provides additional guidance and notes in the implementation of Approvals.

### Approval Table Design

The following tables change in the present invention to support Approvals:

next(underscore)step

\* The apprvl(underscore)ind column indicates whether the next(underscore)step is for an Approval. The mapping tool and the workflow engine handle Approvals slightly differently from other To Dos (see below).

message(underscore)queue(underscore)l

\* The apprvl(underscore)ind column is added to indicate that the To Do is an Approval To Do. The Task Detail window will allow Approval from there if TRUE.

The following tables are added :

apprvl(underscore)enabled(underscore)activity

The apprvl(underscore)enabled(underscore)activity table contains the IDs of Activities that are Approval Enabled. It contains the following columns:

\* activity(underscore)id (primary key) - The ID of an activity that has been coded to support customized Approvals.

\* apprvl(underscore)event(underscore)id (primary key) - The ID of an event which can generate an Approval To Do.

\* reapprvl(underscore)event(underscore)id - The ID of an event which will require reapproval because of changes to the object being approved.

\* apprvl(underscore)activity(underscore)id - The ID of the activity where the Approval should be performed, i.e. the Next Step activity.

\* task(underscore)detail(underscore)apprvl(underscore)id - The name of the PowerBuilder object that contains the application code to be used by Task Detail to support Approvals for this activity.

apprvl(underscore)list

The apprvl(underscore)list table contains Approval List IDs associated with the activities that use them. It contains the following columns:

- \* apprvl(underscore)list(underscore)id (primary key) - The ID of an Approval List.
- \* apprvl(underscore)activity(underscore)id (primary key) - The ID of an activity that has access to this list for Approvals.
- \* notes - Text used as needed.

apprvl(underscore)list(underscore)detail

The apprvl-list(underscore)detail table contains a list of users, groups, roles or structures functions which can be associated with an Approval process. It contains the following columns:

- \* apprvl(underscore)list(underscore)id (primary(underscore)key) - The ID of the Approval List whose members are defined here.
- \* apprvl(underscore)activity(underscore)id (primary key) The activity that has access to this list.
- \* apprvl(underscore)level (primary key) - Indicates the order in which Approvals are performed.
- \* apprvr(underscore)id (primary key) - The user, workgroup or role ID of this approver or the default approver should a structures function fail to be resolved.
- \* apprvr(underscore)type(underscore)code (primary key) - Indicates whether approver(underscore)id is a user, workgroup, role or structure function.
- \* structure(underscore)function(underscore)type - Indicates the type of structure being used.
- \* structure(underscore)group(underscore)id - The ID of the structure group being referenced.
- \* structure(underscore)name - The name of the structure being referenced.
- \* col(underscore)id - The column ID of the data being used to resolve this structure function.
- \* layer(underscore)name - The name of the Layer.
- \* default(underscore)type - Indicates whether approver(underscore)id is a user, workgroup, role. Relevant when the structure function cannot be resolved and the approver(underscore)id is used as a default.

substitute



The Substitute tables contains a list of users who have substitutes defined for them. It contains the following columns:

- \* user(underscore)id (primary key)- The user(underscore)id of the user whose messages will be sent to the substitute.
- \* substitute(underscore)owner(underscore)id - The user id or workgroup id of the substitute user or workgroup.
- \* substitute(underscore)owner(underscore)type - Indicates whether substitute(underscore)id is a user or workgroup.
- \* active(underscore)ind - Indicates whether the substitute is currently being used.
- \* notes - Text used as needed.

tracking/status

The data portion of this table will remain constant while the number and datatype of the keys will vary from application to application.

- \* instance key(s) (primary key)
- \* apprvl(underscore)level (primary key)- Order in which approvals are performed.
- \* apprvr(underscore)id (primary key)- The user, workgroup or role ID of this approver.
- \* apprvr(underscore)type(underscore)code (primary key) - User, Workgroup or Role
- \* apprvl(underscore)status(underscore)code - N = None (hasnt been sent a message to review yet)

P = Pending (has been sent a message but has not responded)

A = Approved

R = Rejected

\* apprvl(underscore)status(underscore)date - Date of the last status change

\* actual(underscore)apprvr(underscore)id - ID of the user who performed the status update.

End User Windows

Activity Window Approve/Reject Standard

The new activity will inherit from the original activity to provide a complete, read-only view of whatever is being Approved. Functionality includes Approve, Reject, Comments, Approval Complete, generation of next level To Do messages and updates to the instance tracking table. The Yellow Sticky object is moveable so that all portions of the original activity can be seen.

#### Task Detail

plt0040(underscore)todo needs to be modified so that when the message queue row indicates that the Next Step activity is an Approval, an additional button will be displayed. When rows from the detail list are selected, enable the button. If Approval button clicked, execute the application provided code to update the Approval Tracking, generate trig(underscore)events and complete the approval process as needed. A reusable PowerBuilder object will contain the necessary application code.

#### User Substitute

The Substitute window allows a user to define a substitute for himself. The key will always be the user's user(underscore)id and will not be able to be modified. The User Substitute window and supporting stored procedures will have to make sure that infinite loops within the Substitutes list are not allowed (User a is a substitute for user B who is a substitute for user A). Default workflow should be delivered with the system so that an informational To Do is generated when a user is made a substitute for another user.

#### Administration and Tracking Windows

##### Approval List Definition

This window allows the administrator to create an Approval List. Each list is keyed by the list name and the activity ID for which it is applicable. A Popmenu and response window support Copy from one List to another.

##### Tracking/Status Ancestor

This window is keyed by the application key(s). It shows the current status of the Approval process for the item shown. It also shows a list of the future approvers and allows the current user to Approve or Reject if appropriate. PopMenu and Tool Bar provide Accept, Reject and Zoom to Document actions.

##### Administrator Substitute

The Administrator Substitute window is a simple tabular window that allows the administrator to change any users substitute. The key is user(underscore)id and the data values are assign(underscore)to type and user or workgroup.

Rule: Add, change, delete of Substitutes does NOT impact existing workflow.

## Workflow Mapping Tool

When an Approval Enabled activity is mapped, the map knows which events are Approval Enabled. When the Insert event is chosen from the windows list of available events, the Approval Activity is presented as a possible next step. When chosen the windows presentation is slightly different in that Approval Lists are presented for the assignment of To Do messages.

### Application Impact Issues

Approval Enabled activities must:

- 1) Have moved Approval Lists and Substitutes to the common Platform tables. Note: Substitutes will now apply to all workflow NOT just Approvals. These tables and their respective maintenance windows will replace the applications' individual implementations. This means that two windows can be dropped for each Approval Enabled activity.
- 2) Use the new TrigEvent function to manage approval processing and to process "Approval Complete" code, if the user has chosen to NOT approve under certain conditions. This function will return an indicator of Approval Complete and the ID of the Approver and its type code which was used to generate the Tracking Table rows. The application window will provide a string containing `svr(underscore)db(underscore)owner.stored(underscore)proc` and the formatted key values to Insert rows into the Tracking table. The workflow engine will provide the User/Workgroup/Role ID and the Approval Level for each approver. Structures functions will be resolved at this time. This means that some code can be removed from an activity window. Each application designer should consider whether it would be useful to store the name of the Approver used.
- 3) Have written the PowerBuilder function which will process Approvals from the Task Detail window. A model will be provided and an attempt will be made to require the bare minimum of code from the applications. Essentially, a fill-in-the-blanks with code that currently exists in other places.

Other steps which could be taken to move in the direction of an internally consistent Approval process are:

- 1) Rewrite Tracking/Status windows to use new standard.
- 2) Use the model for Approve/Reject from the document in addition to the Approve from the Tracking window.

### StreamBuilder Impact Issues

#### Approval Activity window

Users who build activities which they wish to Approval Enable will need a GUI to enter and maintain the `apprvl(underscore)enabled(underscore)activity` table. Approval Enable one of the

existing Sample Application windows. (psa0800(underscore)invoice). Use the new Yellow Sticky to do the Approval on a read-only view of the original activity using the new model. (psa0810(underscore)invoice(underscore)approval).

#### Approval Tracking Window

Provide a sample of the Approval Tracking/Status implementation using the new Ancestor. (psa0850(underscore)invoice(underscore)status).

#### Task Detail Approval object

Provide a sample of the PowerBuilder function used to Approve from the Task Detail window. (psa0800(underscore)invoice(underscore)approve).

#### Dependencies

- 1) Structures integration requires that all structures that are used within workflows be replicated to all work flow servers 110 in the system.
- 2) The use of the UpOne() and UpToLayer() structures functions as the assign to in next step options requires that structures be enhanced include work flow assignee fields within the point definition.
- 3) Structures integration requires that the structures team provide stored procedures to traverse a structure and to validate workflows set up using structures.

#### Workflow Mapping Tool

This Workflow Mapping Tool for the Conditional Workflow System of the present invention is described in further detail below. The new Workflow Mapping Tool, or WMT, sets a new standard for creating and displaying the Conditional Workflows.

The Workflow Workbench of SmartStream 3.0, available from Dun & Bradstreet Software Services, Inc., consists of the Workflow Workbench Definition window and other activity windows such as Activity Definition, Workflow Event Definition, Activities Workflow Event Definition, and Step and Assignments Definition. All the definition windows are accessible from the Workflow Workbench Definition window. These windows present a loosely integrated methodology for defining workflow objects. A primary function of the present invention WMT is to integrate all the workflow Step definition and management functions in order to improve the functionality and maintainability of the SmartStream transaction based workflows. The WMT, is implemented as a separate application. It, however, is tightly integrated with the SmartStream environment. Users invoke it from the SmartStream window. The Activity Definition, Workflow Event Definition, and the Activities Workflow Event Definition windows remain within the SmartStream application but are accessible from the WMT via the Zoom To feature.

The purpose of the Workflow Mapping Tool is to provide workflow developers and administrators with an easy to use, graphical mechanism for developing and maintaining business process workflows. The Workflow Mapping Tool provides the interface by which end users configure the workflow data upon which the SmartStream workflow engine operates. In this way, the WMT allows users to graphically represent and manage the workflow data which controls which controls their various workflow enabled business applications. A major component of this purpose is to allow SmartStream users to configure the DBS provided workflows of the SmartStream applications. Additionally, users who develop their own client/server applications can workflow enable such applications by constructing workflow maps for them through the use of the WMT.

The WMT is a client application which can provide a graphical representation of the workflow definition data for a business. The graphical representation is then used as the access handle for configuring the underlying workflow definition. These graphical workflow drawings are referred to as workflow maps (a.k.a. workpages).

A workflow map, as shown in Figure 58, is essentially a diagram which depicts a Start Step 5801 that begins the workflow; and one or more of the resulting workflow events and steps (5802) that comprise a part of the workflow definition. The boxes on the diagram represent the steps in the workflow. The connecting lines represent the workflow events which facilitate a transition to a subsequent step. A workflow map can display all, or just a portion of the workflow data upon which the workflow engine operates.

Workflow Maps are the fundamental window type in the WMT. They are the primary graphical interface object through which the user accesses and controls the data.

The WMT performs two fundamental tasks via the workflow map: it allows users to browse (display) existing information (workflow steps & events) for an established workflow; and it allows users to construct new workflows or to modify the existing ones. Both of these fundamental tasks are commonly performed in any given usage of the workflow mapping tool. Since the WMT is an MDI (Multiple Document Interface) window, it allows the user to operate on several workflow maps at once. Each workflow map is a child window within the WMT.

The primary object depicted on a workflow map is a Workflow STEP. The WMT defines two basic types of workflow steps: a Start Step 5801; and standard Steps. Standard workflow steps are further broken down into two types: an activity based step, and an information only steps (a.k.a. Informational ToDos, Informational steps). In addition to the steps, a workflow map also manages and displays Workflow Events. Workflow Events are represented by the lines 5803 connecting the steps on the diagram.

The WMT allows the user 120 to organize a set of workflow maps into a workbook. Typically, a workflow map represents only a portion of a workflow, and a workbook contains all of the maps for a particular business process.

The Start Step 5801 establishes an initial context for a workflow map. The Start Step itself does not represent a specific step in the workflow. Moreover, it is used to represent any one of a set of

steps in the system which may be used to arrive at the subsequent steps via one of the Start Steps workflow events. Therefore, the Start Step represents a generalized means by which subsequent steps in the workflow are reached.

Standard workflow steps 5804 represent actual workflow data records in the database. They define subsequent activities to be performed in response to incoming events. They are represented on the workflow maps as (approx.) 1 square with a staircase picture 5805 in the center left portion of the square. Each of the workflow steps represents a complete set of workflow data for the step. This workflow data is referred to as the steps properties. The graphical representation of a step object provides access to a tabbed dialog box through which the property values are established and maintained. The user can access the properties dialog of any workflow object by one of several methods: Selecting the object and using the Show Properties menu selection; Double clicking on the object; or selecting the object and using the Properties menu function of the right mouse menu. The specifics of each objects properties will be discussed later on in this document.

Standard workflow steps (both activity based and informational steps) may be conditional steps. This means that they may have a set of entry conditions associated with them. The entry conditions are a set of expressions that govern whether or not the step will actually be generated (arrived at). A conditional step is represented on the map by a decision diamond graphic 5808 attached to the immediate left of the step. Steps which have no entry conditions are referred to as unconditional steps, while steps having the entry conditions are referred to as conditional steps.

Informational Steps 5806 are very similar to the activity based ,workflow steps. They are also represented on the workflow map as a 1 square, however they contain an I (information) icon. Information steps are essentially a special case of an activity based step whose activity is specified as a standard SmartStream Informational ToDo activity. This activity, as defined in SmartStream, has NO output events and is subsequently often used to terminate a particular flow path in a workflow. This is main reason that Informational steps play such a prominent role in workflow definitions, and the reason they exist as a separate objects within the system.

Workflow Events 5803 are represented by the connecting lines on the workflow map. There are several configurations possible for connecting workflow steps. A single workflow event may connect to one or more subsequent workflow steps. At run-time, a workflow step may generate an specific event form a set of events which it is capable of generating. The WMT can provide a graphical display which shows a flow path for each of the possible events. Since workflows can get arbitrarily complex, a particular map typically depicts a primary flow path of interest in order to avoid graphical clutter.

As previously mentioned, there are several configurations possible for connection workflow steps. The first and simplest construct is a set of steps that are connected sequentially. This construct is represented directly by a workflow event line 5803 connecting the two steps. Such a line may connect two steps directly (unconditional step), or it may connect to a conditional step at the left point of the conditional graphic (diamond) (FIG. 59) When an event connects to a conditional step, a run-time evaluation is made by the SmartStream workflow engine (based on the specified conditions) as to whether or not to generate a task for that particular step.

Additionally, steps may be connected in parallel. (FIG. 60) This type of construct indicates that both steps will be evaluated in response to the single instance of the singular workflow event. Like the serial connection, a parallel connection construct may connect to conditional step. When unconditional steps are connected in parallel, the workflow engine will generate tasks for each of the connected steps. This construct therefore allows parallel flow paths 6001 to exist.

Whenever an events connects to a conditional step, the possibility exists that the conditional expression may evaluate to false. In this case the conditional step will not be executed. If no other steps are connected to the event, the flow will stop. However, a conditional step may have one or more Alternate Steps connected. An alternate steps is connected to the bottom point 6101 of the decision diamond. it (FIG. 61) This connection construct allows the SmartStream workflow engine to continuing evaluating potential target steps to be performed. An alternate step itself may be conditional, and subsequently, it may have its own alternate steps. When the last step in an alternate step sequence is unconditional, it guarantees that at least one step will result in response to the particular workflow event. All of the lines connecting a sequence of alternate steps represent the same workflow event (i.e.: Alternate steps share the same input events as their parent).

The WMT provides a floating tool pallet 6201 as shown in FIG. 62. The tools on the palette are used to build and modify workflow maps. The following shows the palette tools and their function:

To create new steps, the user selects (picks) one of the middle 2 pallet icons and clicks somewhere on the map window to create a new object of that type on the map. Once created in this manner, the new step is yet undefined as it is not connected to an event. Undefined steps are displayed with a yellow step name region, as opposed to the light blue color of defined (connected) steps.

To complete the step definition, the user must have specified the Step Name and Activity properties for the step (via the step properties dialog), and then use the Connection tool (the arrow icon on the pallet) to draw a line from the upstream step to the new downstream step.

Each mapping tool object (Start Step, Step, and Event) has a set of properties associated with the object. These properties control the actual definition of the workflow data for the object, and subsequently its representation on the map. Additionally, each of these objects has an active right mouse button menu associated with it. The right mouse button menu is typically used to access a common set of functions that apply to the selected object. For example, each objects right mouse menu contains an entry to invoke the Properties dialog for the object. The tabbed property sheet provides a one stop shopping mechanism for managing for the object

Referring to FIGS. 63 and 64, most of the actual workflow data is managed via the Properties dialog for each of the workflow objects. The Start Steps properties dialog is a subset of a standard workflow step because is actually a generalized workflow step used to provide the starting context (event) information for the map. The property dialog for the Start Step contains only 2 tabs: the Definition tab 6301 and the Output Events tab 6302, referring to FIG. 63 and FIG. 64 respectively. The Definition tab (FIG. 63) allows the user to select a generalized starting

activity (one of the SmartStream, or user defined, activity windows). The Output Events tab (FIG. 64) then displays the list of possible events for the selected activity. The user may select (check) one of more of these to provide a starting context for the map. If the actual workflow has already been defined (i.e.: it already exists within the database), the WMT will automatically display all of the existing steps connected to each of the selected output events. If there are no defined steps for the selected event (i.e.: no workflow exists), the WMT will display an unconnected event coming out of the Start Step. (Unconnected events display a circle connector at the right end instead of an arrowhead.) Since the output events property is shared by the starting step and all other workflow steps, this methodology affords the user a mechanism for browsing any existing workflow

Referring to FIG. 65, standard workflow steps (both activity based and informational) also have a properties dialog box. This dialog contains the following 5 tabs: Definition 6501; Entry Conditions 6502; Assignment 6503; Input Mapping 6504; and Output Events 6505.

The definitions tab 6501 allows the user to specify the Name of the workflow step; its priority; and its enabled status. The Intl... button 6511 invokes the dialog shown in FIG. 66. It allows the user to specify translated (e.g.: French) versions of the steps name.

Referring to FIG. 67, the entry conditions tab 6502 allow the user to craft a conditional expression based on the data columns which are passed with the steps incoming event. The conditional expression may a compound one (a set of expressions ANDed or ORd together). If the conditional grid is left blank, then the step is defined as being unconditional. This tab will not be active if the step is undefined (unconnected), as there is no incoming event data upon which to base a conditional expression.

There is only one set of entry conditions allowed for a step. These conditional expressions are one of the most important features in the Conditional Workflow system. The entry conditions dialog applies the 32 column values from the incoming Event constructing these expressions. These values are accessible from the dropdowns of the Field column 6701 of FIG. 67. The Operator set is dependent on the type of the data selected in the Field column. For example, if the data type of the field is a integer, the Operator may be equal, greater, less, and etc. The Logical column 6702 contains the boolean operators like AND, OR, and NOT to aggregate expressions into a larger logic construct. If the overall expressions are TRUE, the designated Next Step will be executed. Otherwise the Alternate Next Step will take place. Alternate Steps are optional.

The assignments tab (FIG. 68) allows the user to control which SmartStream user is responsible for performing this steps activity when the step happens. Such assignments may be conditionally computed and are the result of the evaluation of an assignment decision tree by the workflow engine. The assignments table shows a decision tree 6801 as represented by the decision diamond with its starting input at the top and the TRUE evaluation (assignment) shown at the right, the FALSE evaluation exits at the bottom and either enters a new decision diamond (conditional assignment), or encounters a default (unconditional) assignment. The default assignment is always the last one in the tree and is unconditional. It may not be removed. It's value (the actual assignee) may be changed. Unconditional assignments are added and removed by the New/Delete buttons to the right of the tree. They are always inserted just above the



currently selected node. The New... button 6802 invokes the dialog box shown in FIG. 69 which allows the user to specify the conditions 6901 (see the discussion of the Entry Conditions TAB), and the assignee type 6902.

The Input Mapping Tab (FIG. 70) allows the user to manage the mapping of the incoming (event) data into the steps activity window fields. The leftmost column 7001 is fixed and displays the data columns that are bound to the event. The middle column 7002 displays the fields which are to receive (as initial values) the data from corresponding event column. The third column 7003 manages the display order of the event data on the SmartStream Task Details (synopsis) window.

The Output Events tab (FIG. 71) is used to control the visible navigation of the map through the selection of specific output events for the specified activity. It controls the visible portion of the map and does not impact the actual workflow itself. It functions exactly as the Output Events tab of the Start Step does. Specifically in the way that it automatically displays all of the already exiting selected step for a checked event.

Users create new steps by selecting the appropriate icon from the tool palette (FIG. 62). They may select a standard step, or an informational step. Users connect new steps by using the connection tool on the tool palette (e.g., the arrow 6211).

There are two ways in which steps may be removed from the workflow map. The first is to simply remove the graphical picture from the map. This operation is referred to a REMOVING a step from the map. This operation does not affect the underlying workflow data. Indeed, the step remains as a part of the actual workflow. It is only the current visual picture that is altered. The second means of removing steps from a workflow map is to delete them. The delete operation is described below.

Steps may be deleted by selecting the step and using the menu bar delete, or the right mouse menu Delete From Workflow selection. This action is fundamentally different from the Remove From Workflow MAP operation discussed below. Deleting a step from a workflow definition implies removing it from the workflow map.

The visual representation of the map may be altered in two ways. First, by manipulating the scale factor at which the map is drawn. Refer to the Scale menu selection under the View menu. The second method of managing the visual representation is throughout selectively removing workflow steps from the map. This procedure does not delete such steps from the workflow definition (from the database). It simply removes them from the current rendering of the map (presumably because they are un-important to the primary flow path being displayed).

A WorkBook is a collection of workflow maps. There are no restrictions on placing the workflow maps into workbook containers however it is recommended that related maps be placed into a common workbook.

Unlike an existing Workflow Workbench map that may start from any step activities in the flow, a WMT workflow map usually depicts a business process like Loan Application. A business

process may contain many step activities that are insignificant to the entire flow. It is unnecessary to build workflows for all the step activities in the flow. This is a major difference between the WMT and the previous Workflow Workbench maps.

Users can drag an object from the Palette and drop it on a map infers creating the object. Since the WMT is an OLE Server, drag & drop will result in the WMT objects to be copied or moved to the other OLE containers.

The context-sensitive Right Mouse Popup menu is available for all the WMT objects. The popup menu contains frequently used commands and access to the property sheet of the object. The right mouse will also act like an implicit select. For example, if there is no object selected, the right-mouse will select the pointing object and display the popup menu. . For novice users the Right Mouse Menu also provides a list of suggested commands for the object. Depending on the state of an object, the unavailable commands will be grayed.

The WMT is an OLE Server so that a workflow map, can be linked or embedded into another applications container window.

The various items on the horizontal menu 5899 shown in FIG. 58 are described in further detail below.

#### File Menu

- \* New

This command creates a new Workflow Map with the Untitled name. A new workflow map will contain a Starting Activity object on the left-top corner of the map.

- \* Open...

This command displays a dialog box for selecting workflow maps (by workbook container) (see FIG. 72).

- \* Close

This command closes the active workflow map. If the map has modifications, a warning message will be displayed.

- \* Save

Saves the current workflow map..

- \* SaveAs...

This command saves the current Workflow Map using a new name. The Workflow Map name is a 30 characters string. See FIG. 73.

- \* Exit

Exit the Workflow Mapping Tool program. This selection will prompt user to save the modifications if exist.

## Edit Menu

The Edit Menu provides a set of commands to manipulate the workflow objects.

- \* Remove From Workflow Map Only

This command deletes selected objects from a workflow map. It does not remove any of the underlying data from the database.

- \* Delete From Workflow

WMT deletes all the graphical data and workflow data that resides on the next(underscore)step and its normalized tables. If the deleted objects contained multiple connected Next Steps, WMT deletes only the first level Next Step objects' data. The Next Step objects that directly connected to the deleted objects will be marked invalid. Users can detect the difference visually of an invalid object. The activity(underscore)master and event(underscore)master tables remain intact in this operation.

- \* Select All

This command selects all the objects on a workflow map. Many of the Edit commands apply only to a single selection. The Select All command however is very convenient for moving the entire picture around on map.

## View Menu

- \* Scale

This command invokes the dialog box shown in FIG. 74 which allow users to scale a workflow map with the specified factor.

- \* Grid Visible

Grid is a workpage's property. If user selects this option, a check mark will display at the menu.

- \* Snap to Grid

The newly created objects will be snapped to the closest grid on the workflow map. If user selects this option, a check mark will be displayed at the menu

- \* Tool Palette

Tool Palette is a workpage's property. Tool Palette will not display if there is no open workflow map. Since the Workflow Mapping Tool which is a MDI window supports only one type of document, the Tool Palette will be shared among all the WMTs child windows.

User can turn on or off the Tool Palette at the workflow map level. If user selects this option, a check mark will display at the menu. This selection is persistent with the workflow map.

\* Tool Bar

The Tool Bar exists both in the mainframe and map windows. In the mainframe window the Tool Bar contains only the New and Open buttons. All other buttons pertain to the map window are dimmed. In the map window the Tool Bar buttons include Print, Cut, Paste, Alignments and etc.

User can turn on or off the Tool Bar. If selected, a check mark will display at the menu. This selection is persistent with the WMT.

\* Status Bar

The Status Bar exists both in the mainframe and map windows. User can turn on or off the Tool Bar. If selected, a check mark will display at the menu. This selection is persistent with the WMT.

Layout Menu

This Menu provides a set of commands that are purely graphical. None of these operations will affect the workflow data on the database.

\* Align

Various align methods are provided in the pop menu. Users may align objects based on the Tops, Bottoms, Left Sides, and Right Sides of the objects.

\* Align To Grid

This command allows the selected objects be snapped to the closest grid line.

What has been described above are preferred embodiments of the present invention. It is of course not possible to describe every conceivable combination of components or methodologies for purposes of describing the present invention, but one of ordinary skill in the art will recognize that many further combinations and permutations of the present invention are possible. All such possible modifications are to be included within the scope of the claimed invention, as defined by the appended claims below.

**Claims:**

1. A multi-user computer system operative to control the flow of data between a plurality of users of the computer system in connection with the performance of activities by the plurality of users, wherein the activities generate or use the data, the system comprising:

means for receiving first user information input in connection with a first activity;

means for defining a plurality of activities performable by a plurality of users in connection with the performance of the work flow of a particular application;

means for evaluating the first user information; means responsive to the evaluating means for generating a conditional signal; and

means responsive to the conditional signal for routing work flow to another user, wherein the means further includes:

means responsive to a first state of the condition signal for routing the work flow to a second user for performance of a second activity; and

means responsive to a second state of the condition signal for routing the work flow to a third user for performance of a third activity.

2. The system as defined in claim 1, wherein the second user and the third user are identical.

3. The system as defined in claim 1, wherein the second activity is identical to the third activity.

4. The system as defined in claim 1, wherein the first user information input comprises execution of a first activity in the work flow of a selected program.

5. The system as defined in claim 1, wherein the defining means includes at least one table listing a plurality of activities required for execution of a selected work flow application.

6. The system as defined in claim 5, wherein the defining means further includes data defining a sequence for performing the plurality of activities.

7. The system as defined in claim 6, wherein the sequence of the second activity is conditioned upon performance of the first activity.

8. The system as defined in claim 1, wherein the defining means further includes data defining a selected one of the users who must perform at least one of the plurality of activities.

9. The system as defined in claim 8, wherein the selection of an activity for performance by the second user is conditioned upon the performance of the first activity by the first user.

10. The system as defined in claim 8, wherein the at least one activity is performable by more than one user.

11. The system as defined in claim 10, wherein the second and third user alternatively perform the second activity, only after the first user has performed the first activity in the work flow application.

12. The system as defined in claim 11, wherein the defining means is responsive to the performance of the second activity to inhibit repeated performance of the second activity.

13. The system as defined in claim 6, wherein the defining means includes condition data defining the conditions upon which selected activities among the plurality of activities are performed in connection with the work flow of a particular application.

14. The system as defined in claim 13, wherein the condition data defines that the second activity is performable only after the first activity has been performed.

15. The system as defined in claim 13, wherein the condition data defines that the third activity is performable only after the first activity has been performed, but not after the second activity has been performed.

16. The system as defined in claim 13, wherein the condition data defines that the third activity is performable only after the first activity and the second activity have been performed.

17. The system as defined in claim 13, wherein the condition data defines that the second activity is performable only after the first activity has been performed and a subsequent activity has been performed by the third user.

18. The system as defined in claim 6, wherein the defining means includes condition data defining the conditions upon which selected users among the plurality of users may perform selected activities among the plurality of activities that may be performed in connection with the work flow of a particular application.

19. The system as defined in claim 18, wherein the condition data defines that the second activity is performable by the second user only after the first user has performed the first activity.

20. The system as defined in claim 18, wherein the condition data defines that the third activity is performable by the third user only after the first user has performed the first activity and the second activity has been performed.

21. The system as defined in claim 1, wherein the evaluating means is operative to evaluate the defining means.

22. The system as defined in claim 1, wherein the condition signal is indicative of a plurality of next step activities.

23. The system as defined in claim 22, further comprising means responsive to the performance of the first activity by the first user, for activating the condition signal to indicate that both the second activity and the third activity may then be performed.

24. The system as defined in claim 22, further comprising means responsive to the performance of the first activity by the first user, for activating the condition signal to indicate that the second activity may then be performed by the second user.

25. The system as defined in claim 24, wherein the activating means is further operative to activate the condition signal to indicate that the third activity may then be performed by either the second user or the third user.

26. The system as defined in claim 1, wherein the routing means is operative to inhibit performance of a selected activity in a work flow application, until all prerequisite activities have been performed.

27. A computer readable storage medium containing program code for controlling the operation of the system defined in claim 1.

?